NAVAL POSTGRADUATE SCHOOL

MONTEREY, CALIFORNIA

THESIS

NEAREST NEIGHBOR CLASSIFICATION USING A DENSITY SENSITIVE DISTANCE MEASUREMENT

by

Joshua Jeremy Burkholder

September 2009

Thesis Advisor:	Kevin Squire
Second Reader:	Mathias Kolsch

This thesis was done at the MOVES Institute
Approved for public release; distribution is unlimited

REPORT DOCUMENTATION PAGE			Form Approved OMB No. 0704-0188
Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instruction, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188) Washington DC 20503.			
1. AGENCY USE ONLY ***(Leave blank)***	**2. REPORT DATE** September 2009	**3. REPORT TYPE AND DATES COVERED** Master's Thesis	
4. TITLE AND SUBTITLE Nearest Neighbor Classification Using A Density Sensitive Distance Measurement		**5. FUNDING NUMBERS**	
6. AUTHOR(S) Joshua Jeremy Burkholder			
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Naval Postgraduate School Monterey, CA 93943-5000		**8. PERFORMING ORGANIZATION REPORT NUMBER**	
9. SPONSORING /MONITORING AGENCY NAME(S) AND ADDRESS(ES) N/A		**10. SPONSORING/MONITORING AGENCY REPORT NUMBER**	
11. SUPPLEMENTARY NOTES The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.			
12a. DISTRIBUTION / AVAILABILITY STATEMENT Approved for public release; distribution is unlimited		**12b. DISTRIBUTION CODE**	

13. ABSTRACT (maximum 200 words)

This work proposes a density sensitive distance measurement that takes into account the density of an underlying dataset to better represent the shape of the data when measuring distance. Kernel density estimation, using kernel bandwidths determined by k -nearest neighbor distances, is used to approximate the density of the underlying dataset. A scale is applied to the resulting kernel density estimate and a line integral is performed along its surface resulting in a density sensitive distance. This work tests the utility of the proposed density sensitive distance measurement using supervised learning. k -Nearest Neighbor classification using both the proposed density sensitive distance measurement and Euclidean distance are compared on the Wisconsin Diagnostic Breast Cancer dataset and the MNIST Database of Handwritten Digits. For perspective, these classifiers are also compared to Support Vector Machine and Random Forests classifiers. Stratified 10-fold cross validation is used to determine the generalization error of each classifier. In all comparisons, k -Nearest Neighbor classification using the proposed density sensitive distance measurement had less generalization error than k -Nearest Neighbor classification using Euclidean distance. For the MNIST dataset, k -Nearest Neighbor classification using the density sensitive distance measurement also had less generalization error than both Support Vector Machine and Random Forests classification.

14. SUBJECT TERMS Classification, Supervised Learning, k-Nearest Neighbor Classification, Euclidean Distance, Mahalanobis Distance, Density Sensitive Distance, Parzen Windows, Manifold Parzen Windows, Kernel Density Estimation			**15. NUMBER OF PAGES** 121
			16. PRICE CODE
17. SECURITY CLASSIFICATION OF REPORT Unclassified	**18. SECURITY CLASSIFICATION OF THIS PAGE** Unclassified	**19. SECURITY CLASSIFICATION OF ABSTRACT** Unclassified	**20. LIMITATION OF ABSTRACT** UU

NSN 7540-01-280-5500

Standard Form 298 (Rev. 2-89)
Prescribed by ANSI Std. 239-18

THIS PAGE INTENTIONALLY LEFT BLANK

Approved for public release; distribution is unlimited

NEAREST NEIGHBOR CLASSIFICATION USING A DENSITY SENSITIVE DISTANCE MEASUREMENT

Joshua Jeremy Burkholder
Lieutenant Commander, United States Navy
B.S., Arizona State University, 1998

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN MODELING, VIRTUAL ENVIRONMENTS, AND SIMULATIONS (MOVES)

from the

NAVAL POSTGRADUATE SCHOOL
September 2009

Author: Joshua Jeremy Burkholder

Approved by: Kevin Squire, Ph.D.
Thesis Advisor

Mathias Kolsch, Ph.D.
Second Reader

Mathias Kolsch, Ph.D.
Chair, MOVES Academic Committee

THIS PAGE INTENTIONALLY LEFT BLANK

ABSTRACT

This work proposes a density sensitive distance measurement that takes into account the density of an underlying dataset to better represent the shape of the data when measuring distance. Kernel density estimation, using kernel bandwidths determined by k-nearest neighbor distances, is used to approximate the density of the underlying dataset. A scale is applied to the resulting kernel density estimate and a line integral is performed along its surface resulting in a density sensitive distance. This work tests the utility of the proposed density sensitive distance measurement using supervised learning. k-Nearest Neighbor classification using both the proposed density sensitive distance measurement and Euclidean distance are compared on the Wisconsin Diagnostic Breast Cancer dataset and the MNIST Database of Handwritten Digits. For perspective, these classifiers are also compared to Support Vector Machine and Random Forests classifiers. Stratified 10-fold cross validation is used to determine the generalization error of each classifier. In all comparisons, k-Nearest Neighbor classification using the proposed density sensitive distance measurement had less generalization error than k-Nearest Neighbor classification using Euclidean distance. For the MNIST dataset, k-Nearest Neighbor classification using the density sensitive distance measurement also had less generalization error than both Support Vector Machine and Random Forests classification.

THIS PAGE INTENTIONALLY LEFT BLANK

TABLE OF CONTENTS

LIST OF FIGURES

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF TABLES

THIS PAGE INTENTIONALLY LEFT BLANK

EXECUTIVE SUMMARY

This work proposes a density sensitive distance measurement that takes into account the density of an underlying dataset to better represent the shape of the data when measuring distance. Kernel density estimation, using kernel bandwidths determined by k-nearest neighbor distances, is used to approximate the density of the underlying dataset. A scale is applied to the resulting kernel density estimate and a line integral is performed along its surface resulting in a density sensitive distance. This work tests the utility of the proposed density sensitive distance measurement using supervised learning. k-Nearest Neighbor classification using both the proposed density sensitive distance measurement and Euclidean distance are compared on the Wisconsin Diagnostic Breast Cancer dataset and the MNIST Database of Handwritten Digits. For perspective, these classifiers are also compared to Support Vector Machine and Random Forests classifiers. Stratified 10-fold cross validation is used to determine the generalization error of each classifier. In all comparisons, k-Nearest Neighbor classification using the proposed density sensitive distance measurement had less generalization error than k-Nearest Neighbor classification using Euclidean distance. For the MNIST dataset, k-Nearest Neighbor classification using the density sensitive distance measurement also had less generalization error than both Support Vector Machine and Random Forests classification.

THIS PAGE INTENTIONALLY LEFT BLANK

ACKNOWLEDGMENTS

I would like to acknowledge my wife, Shari Burkholder, who has put up with my schedule for these last couple of years and has kept things moving forward when I seemed to do everything in my power to keep them put.

I would also like to acknowledge my son, Sebastian Burkholder, who gave me most of the ideas in this thesis as I watched him grow from age six months to two and a half years of age.

I would finally like to acknowledge my advisor, Kevin Squire, Ph.D., who put up with my inability to turn anything in on time. His tolerance is amazing. Moreover, his ability is to describe machine learning to me in a way that I could actually understand has inspired me more than I can express.

THIS PAGE INTENTIONALLY LEFT BLANK

I. INTRODUCTION

When Operations Specialists, Combat Information Center Watch Officers, or Tactical Action Officers sit at their respective consoles, they often monitor sensor sweeps for incoming and outgoing surface, sub-surface, and aerial traffic. With the help of these watchstanders, a ship's combat system will interpret the sweeps and produce surface, sub-surface, or aerial tracks. If the ship's combat system is too sensitive, then many false tracks are produced. If the ship's combat system is too indifferent, then tracks that should be produced are not. Either way, the majority of a watchstander's time can be spent analyzing whether or not a track in the combat system is actually there and cleaning up tracks that are not. Moreover, since these tracks represent friendly, neutral, or hostile entities, a great amount of care is taken to ensure that tracks are classified correctly. Tracks are analyzed not only for their existence, but also for their operating characteristics and signatures. Since a Combat Information Center would not want to fire on a friendly force, a commercial airliner, a fishing boat, or a cargo ship, a great amount of time is taken to make sure that a hostile track is actually a hostile track. The time taken to verify that the system is correct is necessary because the algorithms in use are noisy. If an anti-ship cruise missile, a low-slow flyer, or an explosives-filled wooden fishing vessel were inbound, then the watchstanders in that Combat Information Center may only have a few seconds from detection to reaction in order to avoid being hit. There is not enough time to verify that something inbound is real and correctly classified. Therefore, new or revised classification algorithms must be employed.

The analysis of the sensor sweeps performed by the combat system falls into the category of computer vision – an application of machine learning. The algorithms used by the combat system are designed to classify the information in the sweeps.

The combat system needs to know how to distinguish a plane from the sky using a three dimensional radar, a ship from the sea using a surface radar, and a submarine from the ocean's floor using sonar without having any understanding of what a plane, the sky, a ship, the sea, a submarine, or the ocean's floor is. To run efficiently (and hopefully

effectively), the combat system simply needs to know that a generic difference exists and how to take advantage of that difference to classify these entities. This difference is often expressed as a simple distance measurement. Since the combat system represents the physical environment as data in some information space, the farther apart two data points are that space, the less likely the corresponding physical objects are related.

Once the combat system is given information, (such as a processed radar feed), the system can pass this information on to the classification algorithms. These algorithms take in the unknown information and efficiently attempt to determine if that information represents a plane, the sky, a ship, the sea, a submarine, or the ocean's floor. Since the classification algorithms have been trained to recognize planes, there is a good chance that information representing a plane will end up closer to where the previous information about planes has ended up. Moreover, a simple distance metric is often used to determine which class an unknown piece of information belongs to. If the information is closer to planes than it is to submarines, then that information is probably a plane.

With this classification in hand, the combat system can perform a variety of cross-referencing to determine if that newly classified item is a friendly, a neutral, a hostile, or simply part of the background.

The first step in any of these classification routines is to receive processed sensor feeds, vice raw sensor feeds. More often than not, raw sensor feeds give little to no relevant information that a classification algorithm would need to do its job. A raw sensor feed usually gives nothing more than a direct reading, not how that reading differs or works in conjunction with previous readings. This is where processing comes into play. A raw sensor feed can be manipulated in order to emphasize invariant aspects of the object of interest. Like processing an image from a digital camera, noise can be eliminated or minimized, changes in intensity can be determined, and normalization can be performed. This information can be included into the processed feed that classification routines receive in order to increase the likelihood of a correct classification.

As previously noted, a distance metric is commonly used in classification algorithms in order to determine how similar or different one thing is from another in the information space. By far, the most common distance measurement utilized by these algorithms is Euclidean distance. Euclidean distance assumes that two data points are similar based on their proximity to each other, without regard to the other things around them. In other words, if we needed to determine that something is a ship or the sea in the information space, then Euclidean distance would simply determine which was closer, without regard to the density of the ships or the sea. We could, however, take into account the density of ships and the sea in order to achieve a density-sensitive distance measurement. In other words, if the ships were the red dots, the seas were the blue dots, and a black dot, equally far away from either the ships or the seas, was a new piece of information that we just received (as in Figure 1), then Euclidean distance would arbitrarily classify that new piece of information because the densities of the classes being dealt with are not taken into account.

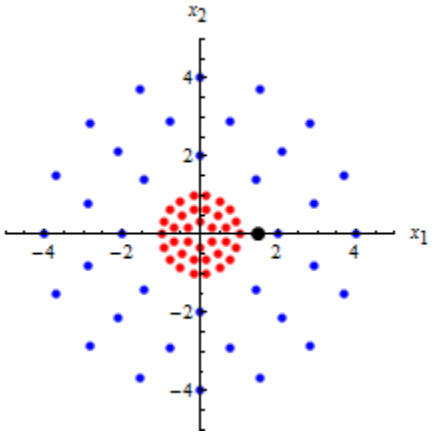


Figure 1. A piece of information (the black dot) equally far away from two classes with different densities (the blue and red dots).

Since the seas (the blue dots) are much more dispersed than the ships (the red dots) in the information space above, then it can be argued that in order for the new piece of information (the black dot) to be considered a ship (red), it ought to be as close to the rest of the ships as all the ships are to each other. In other words, for this new piece of information (the black dot) to be classified as a ship (red), it ought to mimic the level of

dispersion of the previously encountered ships (the red dots). Moreover, since the position of this new information (the black dot) in the information space is more consistent with the density of the seas (the blue dots), then it makes more sense to classify this new information as the sea (blue).

Therefore, we need a distance that is sensitive to the density of the data over which it will measure. Moreover, this density sensitive distance should be able to take measurements over any set or subset of data, regardless of class.

II. RELATED WORK

The creation of this density sensitive distance measurement will be based on previous work regarding kernel density estimates, distance measurements, and principal component analysis.

A. KERNEL DENSITY ESTIMATES

Although there are many different kernel density estimates, the key ones that this work most relied on are Parzen Windows and Manifold Parzen Windows.

1. Parzen Windows (Parzen, 1962)

Parzen Windows is a method of estimating the probability density function $f(x)$ of a random variable X from sample data $(x_1,\ldots,x_m)$ generated by that random variable. The Parzen Windows method centers a weighting function K with a common width h on top of each sample x_i in the dataset and then adds up a scalar multiple of all those functions to produce an estimate of the underlying probability density function $\hat{f}_m(x)$. Thus, the univariate Parzen Windows estimate of the probability density function is

$$\hat{f}_m(x) = \frac{1}{m}\sum_{i=1}^{m}\frac{1}{h}K\left(\frac{x-x_i}{h}\right)$$

where m is the number of points in the sample dataset, $\hat{f}_m(x)$ is an estimate of the probability density function $f(x)$ using m samples, h is the common width of the weighting function K, x_i is the i-th point in the sample dataset for $i=1,\ldots,m$, and K is the weighting function. For a variety of reasons, the weighting function K must satisfy:

$$\int_{-\infty}^{\infty}K(x)dx = 1.$$

The weighting function K, also called the kernel function, can take many forms; however, one of the more popular weighting functions is the following:

$$K\left(x\right) = \frac{e^{\left(-\frac{x^2}{2}\right)}}{\sqrt{2\pi}}$$

This weighting function satisfies $\int_{-\infty}^{\infty} K\left(x\right)dx = 1$ such that $\hat{f}_m\left(x\right)$ becomes

$$\begin{aligned}\hat{f}_m\left(x\right) &= \frac{1}{m}\sum_{i=1}^{m}\frac{1}{h}K\left(\frac{x-x_i}{h}\right)\\ &= \frac{1}{m}\sum_{i=1}^{m}\frac{1}{h}\left(\frac{e^{\left(-\frac{\left(\frac{x-x_i}{h}\right)^2}{2}\right)}}{\sqrt{2\pi}}\right)\\ &= \frac{1}{m}\sum_{i=1}^{m}\frac{e^{\left(-\frac{\left(x-x_i\right)^2}{2h^2}\right)}}{\sqrt{2\pi}h}\end{aligned}$$

which is the normalized sum of univariate normal distribution probability density functions where x_i is the mean of the i-th function and h is the common standard deviation.

For multivariate Parzen Windows where widths are allowed to vary along each dimension, the estimate of the probability density function becomes

$$\hat{f}_m\left(\vec{\mathbf{x}}\right) = \frac{1}{m}\sum_{i=1}^{m}\left(\prod_{j=1}^{n}\frac{1}{h_j}K\left(\frac{x_j - x_j^{(i)}}{h_j}\right)\right)$$

where m is the number of samples in the dataset, $\vec{\mathbf{x}}$ is an n dimensional variable with components x_j for $j = 1,\ldots,n$, $\hat{f}_m\left(\vec{\mathbf{x}}\right)$ is an estimate of the probability density function $f\left(\vec{\mathbf{x}}\right)$ using m samples of n dimensions, h_j is the width of the weighting function K that is common along the j-th dimension, $x_j^{(i)}$ is the j-th component of i-th sample

from the dataset for $i = 1,\ldots,m$ and $j = 1,\ldots,n$, and K is the weighting function (Wasserman, 2007).

Moreover, if the widths are allowed to covary along all dimensions, then the Parzen Window estimate is

$$\hat{f}_m\left(\vec{\mathbf{x}}\right) = \frac{1}{m}\sum_{i=1}^{m}\frac{1}{h^n}K_{\vec{\vec{\Sigma}}}\left(\frac{\vec{\mathbf{x}} - \vec{\mathbf{x}}_i}{h}\right)$$

(Alpaydin, 2004). For similar reasons as the univariate case, the multivariate weighting function K must satisfy the following:

$$\int_{-\infty}^{\infty}\cdots\int_{-\infty}^{\infty}K\left(\vec{\mathbf{x}}\right)dx_1\cdots dx_n = 1.$$

When $h = 1$, the multivariate weighting function $K_{\vec{\vec{\Sigma}}}$ can be the following:

$$K_{\vec{\vec{\Sigma}}}\left(\vec{\mathbf{x}}\right) = \frac{1}{\left(2\pi\right)^{\frac{n}{2}}\left|\vec{\vec{\Sigma}}\right|^{\frac{1}{2}}}e^{\left(-\frac{1}{2}\vec{\mathbf{x}}^T\vec{\vec{\Sigma}}^{-1}\vec{\mathbf{x}}\right)}$$

where n is the dimension of the variable $\vec{\mathbf{x}}$, $\vec{\vec{\Sigma}}$ is the $n \times n$ covariance matrix of the sample data, $\left|\vec{\vec{\Sigma}}\right|$ is the determinant of that covariance matrix, and $\vec{\vec{\Sigma}}^{-1}$ is the inverse of that covariance matrix. Note that here $\vec{\mathbf{x}}$ is assumed to be a $n \times 1$ column vector such that the transpose of that column vector $\vec{\mathbf{x}}^T$ results in a $1 \times n$ row vector. This multivariate weighting function satisfies $\int_{-\infty}^{\infty}\cdots\int_{-\infty}^{\infty}K\left(\vec{\mathbf{x}}\right)dx_1\cdots dx_n = 1$ such that $\hat{f}_m\left(x\right)$ becomes

$$\hat{f}_m\left(\vec{\mathbf{x}}\right) = \frac{1}{m}\sum_{i=1}^{m}\frac{1}{\left(2\pi\right)^{\frac{n}{2}}\left|\vec{\vec{\Sigma}}\right|^{\frac{1}{2}}}e^{\left(-\frac{1}{2}\left(\vec{\mathbf{x}}-\vec{\mathbf{x}}_i\right)^T\vec{\vec{\Sigma}}^{-1}\left(\vec{\mathbf{x}}-\vec{\mathbf{x}}_i\right)\right)}$$

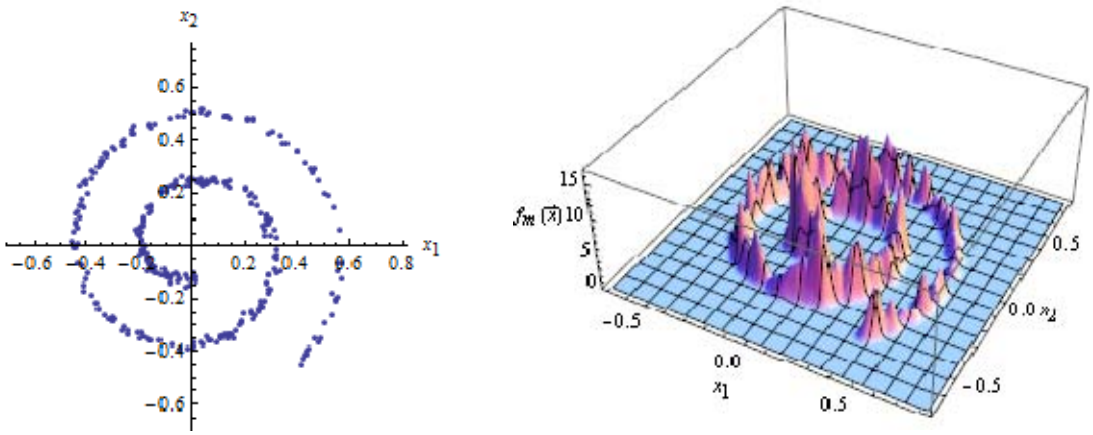


Figure 2. An example of Parzen Windows.

Figure 2 is an example in Parzen Windows (Vincent & Bengio, 2002). On the left, we have a sample dataset. On the right, we have an estimate $f_m\left(\vec{\mathbf{x}}\right)$ of the underlying probability density function using the multivariate weighting function $K_{\vec{\vec{\Sigma}}}$ with diagonal covariance matrix $\vec{\vec{\Sigma}}$ with equal variance in all dimensions (i.e., a "spherical" bivariate normal).

For Parzen Windows, one should recognize that the same width h is used for every weighting function K that is placed on top of each sample in the dataset. Once the width h is determined, it remains constant throughout the rest of the Parzen Windows method; hence, the width h is derived from information that is global to the entire dataset, vice locally adapting to the sample data. This makes the width h difficult to determine for some datasets. For those troublesome datasets, h may be too wide at certain positions in the dataset and too narrow at other positions in the same dataset to accurately estimate the true density of the sample dataset.

Parzen Windows using diagonal covariance matrix $\vec{\vec{\Sigma}}$ with equal variance in all dimensions (i.e., "spherical" multivariate Normal) is the kernel density estimate that will be used for the proposed density sensitive distance measurement.

2. Manifold Parzen Windows (Vincent & Bengio, 2002)

Manifold Parzen Windows assumes that a sample dataset is produced from a lower dimensional manifold. Locally adapting weighting functions K_i are used to estimate that manifold. Manifold Parzen Windows infers the local direction of the underlying manifold by calculating the eigenvalues (i.e., variances) and corresponding eigenvectors (i.e., the directions of variance) associated with the covariance $\vec{\vec{\Sigma}}_i$ of a neighborhood around each sample point. Larger variances are assumed to be associated with the directions tangent to the manifold in order to account for the manifold. Smaller variances are assumed to be associated with the directions normal to the manifold in order to account for the noise off the manifold. A weighting function K_i using that local covariance matrix $\vec{\vec{\Sigma}}_i$ is then placed over the sample point $\vec{\mathbf{x}}_i$; hence, a different local covariance matrix $\vec{\vec{\Sigma}}_i$ is used for each sample $\vec{\mathbf{x}}_i$.

For Manifold Parzen Windows, the neighborhood used to calculate the local covariance matrix for a given sample point can be a hard k-neighborhood, vice a range. In other words, we can compute the local covariance matrix $\vec{\vec{\Sigma}}_i$ associated with a sample point $\vec{\mathbf{x}}_i$ by considering the k-nearest neighbors of that sample point. However, if k is less than the dimension of our data (i.e., $k < n$) or if the k-nearest neighbors do not span the dimension of our data (i.e., the k-nearest neighbors exist in a subspace of our n-dimensional dataset), then the resulting covariance matrix $\vec{\vec{\Sigma}}_i$ would be singular and not invertible. Therefore, an epsilon ε of variance is also added along each dimension in order to guarantee non-singularity and invertibility.

Lastly, we can reduce some of the computational complexity of Manifold Parzen Windows by only considering the estimated dimension of the underlying manifold. The dimension of the underlying manifold can be estimated by only considering the eigenvectors associated with the largest eigenvalues of each local covariance matrix $\vec{\vec{\Sigma}}_i$.

These eigenvectors are estimates of the principal directions of the local manifold. Moreover, the eigenvectors associated with the smallest eigenvalues of each local covariance matrix $\vec{\vec{\Sigma}}_i$ are the directions of noise off the manifold. Therefore, we can discard the eigenvectors of noise, retain our eigenvectors of principal direction, and reduce our local covariance matrices $\vec{\vec{\Sigma}}_i$ down to a dimension that better accounts for the manifold producing our data.

Therefore, a Manifold Parzen Window estimate of the probability density function using a locally adapting multivariate normal weighting function $N_{\vec{\mu}_i, \vec{\vec{\Sigma}}_i}$ is:

$$f_m\left(\vec{\mathbf{x}}\right) = \frac{1}{m}\sum_{i=1}^{m} N_{\vec{\mu}_i, \vec{\vec{\Sigma}}_i}\left(\vec{\mathbf{x}}\right)$$

where m is the number of points in the sample dataset, $f_m\left(\vec{\mathbf{x}}\right)$ is an estimate of the probability density function $f\left(\vec{\mathbf{x}}\right)$ using m samples, $\vec{\boldsymbol{\mu}}_i$ is the i-th point in the sample dataset for $i = 1, \ldots, m$ (previously called $\vec{\mathbf{x}}_i$), $\vec{\vec{\Sigma}}_i$ is the i-th locally calculated covariance matrix, and $N_{\vec{\mu}_i, \vec{\vec{\Sigma}}_i}$ is the locally adapting multivariate normal weighting function defined as

$$N_{\vec{\mu}_i, \vec{\vec{\Sigma}}_i}\left(\vec{\mathbf{x}}\right) = \frac{1}{\left(2\pi\right)^{\frac{n}{2}} \left|\vec{\vec{\Sigma}}_i\right|^{\frac{1}{2}}} e^{\left(-\frac{1}{2}\left(\vec{\mathbf{x}}-\vec{\mu}_i\right)^T \vec{\vec{\Sigma}}_i^{-1}\left(\vec{\mathbf{x}}-\vec{\mu}_i\right)\right)}$$

where $N_{\vec{\mu}_i, \vec{\vec{\Sigma}}_i}$ is defined similarly to the multivariate normal weighting function in the Parzen Windows section.

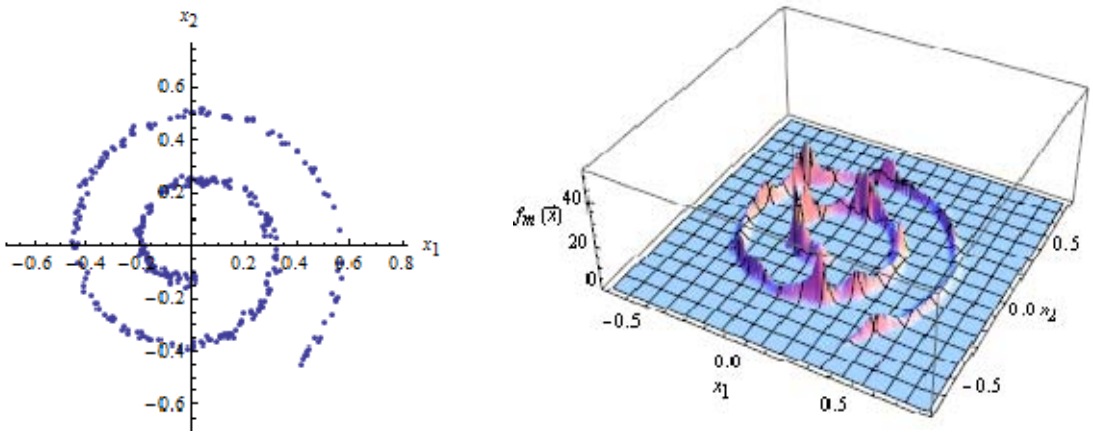


Figure 3. An example of Manifold Parzen Windows.

Figure 3 shows an example in Manifold Parzen Windows from (Vincent & Bengio, 2002). On the left, we have a sample dataset. On the right, we have an estimate $f_m\left(\vec{\mathbf{x}}\right)$ of the underlying probability density function using $N_{\vec{\mu}_i,\vec{\vec{\Sigma}}_i}\left(\vec{\mathbf{x}}\right)$ with local covariance matrices $\vec{\vec{\Sigma}}_i$ calculated using the 10-nearest neighbors of each sample point. Figure 3 should be compared to Figure 2 in order to see how Manifold Parzen Windows differ from Parzen Windows.

For the proposed density sensitive distance measurement, we will use k-th nearest neighbor distances to determine the bandwidth of the kernel used in the density estimate of a dataset similar to the way that Manifold Parzen Windows used the k-nearest neighbors to determine each local covariance matrix.

B. DISTANCE MEASUREMENTS

The proposed density sensitive distance measurement will be designed to be a locally weighted Euclidean distance, one of the Minkowski distances. Moreover, the aspects of other distance measurements, namely the Mahalanobis distance and Wang et al.'s Density Sensitive Distance Metric (Wang, et al., 2006), will also impact this density sensitive distance measurement.

1. Minkowski Distances

The Minkowski distance metrics include the Manhattan, Euclidean, and Chebyshev distances (Zezula, 2006). The generic form of the Minkowski distance metric is the following:

$$\text{distance}_p\left(\vec{\mathbf{x}}^{(o)}, \vec{\mathbf{x}}^{(f)}\right) = \left(\sum_{i=1}^{n} \left| x_i^{(f)} - x_i^{(o)} \right|^p \right)^{\frac{1}{p}}$$

where $p \in \mathbb{R}$ is the power of the metric, $\vec{\mathbf{x}}^{(o)}$ is the initial point (the source point), $\vec{\mathbf{x}}^{(f)}$ is the final point (the destination point), and n is the shared dimension of the points.

a. Manhattan Distance (City-Block Distance)

Manhattan distance takes the following form:

$$\begin{aligned}\text{distance}_1\left(\vec{\mathbf{x}}^{(o)}, \vec{\mathbf{x}}^{(f)}\right) &= \left(\sum_{i=1}^{n} \left| x_i^{(f)} - x_i^{(o)} \right|^1 \right)^{\frac{1}{1}} \\ &= \left| x_1^{(f)} - x_1^{(o)} \right| + \cdots + \left| x_n^{(f)} - x_n^{(o)} \right|\end{aligned}$$

(Zezula, 2006) and has the unit circle detailed in Figure 4; hence, this metric is not invariant to rotation (Samet, 2006).

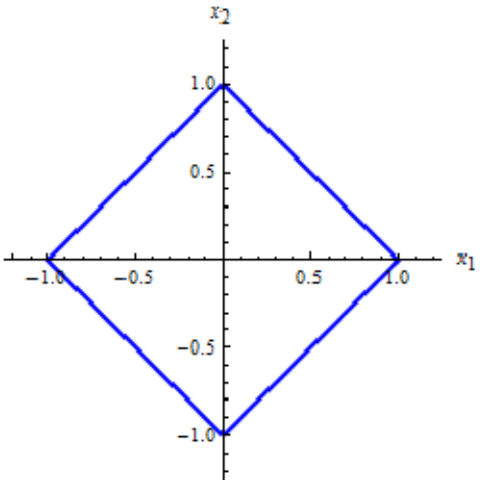


Figure 4. The unit circle of Manhattan distance.

b. Euclidean Distance

Euclidean distance takes the following form:

$$\begin{aligned}\text{distance}_2\left(\vec{\mathbf{x}}^{(o)},\vec{\mathbf{x}}^{(f)}\right) &= \left(\sum_{i=1}^{n}\left|x_i^{(f)}-x_i^{(o)}\right|^2\right)^{\frac{1}{2}} \\ &= \sqrt{\left(x_1^{(f)}-x_1^{(o)}\right)^2+\cdots+\left(x_n^{(f)}-x_n^{(o)}\right)^2} \\ &= \sqrt{\left(\vec{\mathbf{x}}^{(f)}-\vec{\mathbf{x}}^{(o)}\right)^T\left(\vec{\mathbf{x}}^{(f)}-\vec{\mathbf{x}}^{(o)}\right)}\end{aligned}$$

(Zezula, 2006) and has the unit circle detailed in Figure 5; hence, Euclidean distance is invariant to rotation (Samet, 2006).

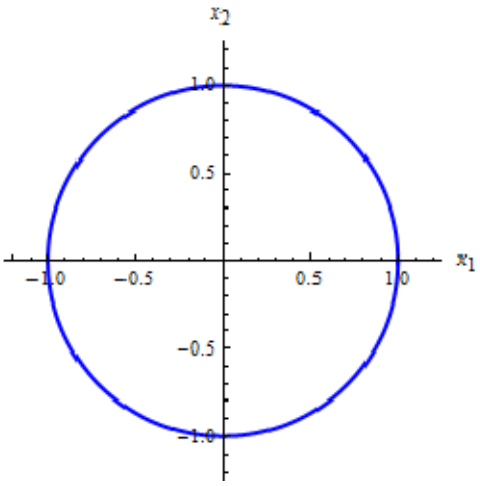


Figure 5. The unit circle of Euclidean distance.

Moreover, since the linearly interpolation from $\vec{\mathbf{x}}^{(o)}$ to $\vec{\mathbf{x}}^{(f)}$ is $\vec{\mathbf{x}}_{o\to f}\left(t\right)=\left(1-t\right)\vec{\mathbf{x}}^{(o)}+\left(t\right)\vec{\mathbf{x}}^{(f)}$ where $0\le t\le 1$ and $d\vec{\mathbf{x}}_{o\to f}\left(t\right)/dt=\vec{\mathbf{x}}^{(f)}-\vec{\mathbf{x}}^{(o)}$, then Euclidean distance can be rewritten as the following:

$$\text{distance}_2\left(\vec{\mathbf{x}}^{(o)},\vec{\mathbf{x}}^{(f)}\right)=\sqrt{\left(d\vec{\mathbf{x}}_{o\to f}\left(t\right)/dt\right)^T\left(d\vec{\mathbf{x}}_{o\to f}\left(t\right)/dt\right)}$$

c. Chebyshev Distance (Chessboard Distance)

Chebyshev distance has the following form:

$$\begin{aligned}\text{distance}_{\infty}\left(\vec{\mathbf{x}}^{(o)},\vec{\mathbf{x}}^{(f)}\right) &= \lim_{p\to\infty}\left(\sum_{i=1}^{n}\left|x_i^{(f)}-x_i^{(o)}\right|^p\right)^{\frac{1}{p}} \\ &= \max\left\{\left|x_1^{(f)}-x_1^{(o)}\right|,\ldots,\left|x_n^{(f)}-x_n^{(o)}\right|\right\}\end{aligned}$$

(Zezula, 2006) and has the unit circle detailed in Figure 6; hence, Chebyshev distance is not invariant to rotation (Samet, 2006).

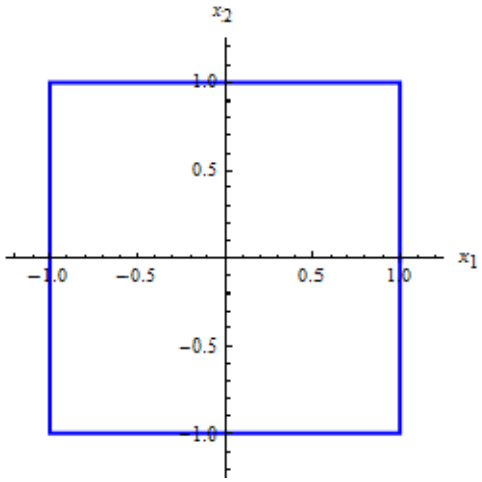


Figure 6. The unit circle of Chebyshev distance.

None of the Minkowski distances take the underlying dataset into account when performing their metric; hence, the distance looks at the dataset uniformly. Moreover, the Minkowski distances do not offer different weights to different pairs of points; hence, these Minkowski distances are not locally weighted.

2. Mahalanobis Distance (Mahalanobis, 1936)

One reaction to the dataset-independent, unity-weighted Euclidean distance is Mahalanobis distance. Mahalanobis distance takes into account the global covariance $\vec{\vec{\Sigma}}$ of a dataset and weights each distance based on this covariance. Mahalanobis distance takes the following form:

$$\text{distance}_{\text{Mahalanobis}}\left(\vec{\mathbf{x}}^{(o)},\vec{\mathbf{x}}^{(f)}\right)=\sqrt{\left(\vec{\mathbf{x}}^{(f)}-\vec{\mathbf{x}}^{(o)}\right)^{T}\vec{\vec{\mathbf{\Sigma}}}^{-1}\left(\vec{\mathbf{x}}^{(f)}-\vec{\mathbf{x}}^{(o)}\right)}$$

and has data dependent unit circles detailed in Figure 7.

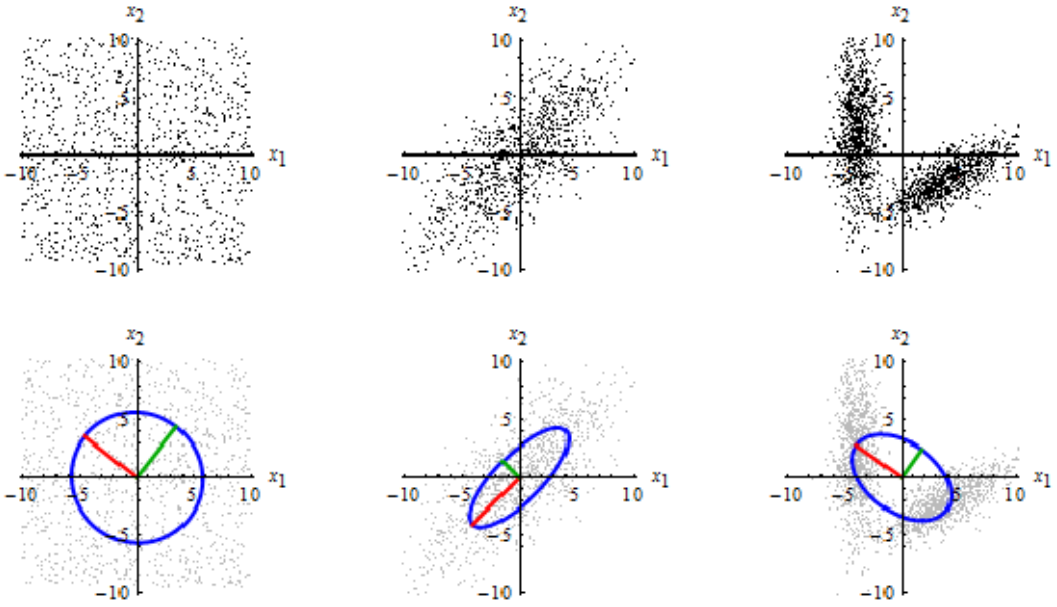


Figure 7. The data dependent unit circles of Mahalanobis distance.

For Mahalanobis distance, the unit for the unit circle is one standard deviation in the direction of each principal component; hence, the unit circle for Mahalanobis is a data dependent ellipsoid whose radii are one standard deviation in each of the principal component directions.

While Mahalanobis distance is data dependent and takes into account the global variance of the dataset, it does not take into account the local densities of the dataset. Moreover, Mahalanobis distance offers no advantage over Euclidean distance when the global variances in each of the principal directions are equivalent (as in the left of Figure 7).

3. Density Sensitive Distance Metric (Manifold Distance) (Wang et al., 2006)

The Density Sensitive Distance Metric of Ling Wang, Liefeng Bo, and Licheng Jiao has the following form:

Let data points be the nodes of graph $G = \left(V, E\right)$ and $p \in V^{\ell}$ be a path of length $\ell = \left|p\right|$ connecting the initial point $\vec{\mathbf{x}}^{(o)} = \vec{\mathbf{x}}^{(1)}$ to the final point $\vec{\mathbf{x}}^{(f)} = \vec{\mathbf{x}}^{(|p|)}$ in which $\left(\vec{\mathbf{x}}^{(k)}, \vec{\mathbf{x}}^{(k+1)}\right) \in E$ for $1 \leq k \leq \left|p\right|$. Let $P_{\vec{\mathbf{x}}^{(o)}, \vec{\mathbf{x}}^{(f)}}$ denote the set of all paths connecting the initial point $\vec{\mathbf{x}}^{(o)}$ to the final point $\vec{\mathbf{x}}^{(f)}$. The density sensitive distance metric between two points is defined to be

$$\operatorname{distance}_{\substack{\text{density}\\\text{sensitive}}}\left(\vec{\mathbf{x}}^{(o)}, \vec{\mathbf{x}}^{(f)}\right) = \min_{p \in P_{\vec{\mathbf{x}}^{(o)}, \vec{\mathbf{x}}^{(f)}}} \sum_{k=1}^{|p|-1} \operatorname{distance}_{\substack{\text{density}\\\text{adjusted}\\\text{length}}}\left(\vec{\mathbf{x}}^{(k)}, \vec{\mathbf{x}}^{(k+1)}\right)$$

where the density adjusted length of a line segment is defined to be

$$\operatorname{distance}_{\substack{\text{density}\\\text{adjusted}\\\text{length}}}\left(\vec{\mathbf{x}}^{(k)}, \vec{\mathbf{x}}^{(k+1)}\right) = \rho^{\operatorname{distance}_2\left(\vec{\mathbf{x}}^{(k)}, \vec{\mathbf{x}}^{(k+1)}\right)} - 1$$

where $\rho \in \mathbb{R}$ is the flexing factor for $\rho > 1$ and $\operatorname{distance}_2\left(\vec{\mathbf{x}}^{(k)}, \vec{\mathbf{x}}^{(k+1)}\right)$ is Euclidean distance between $\vec{\mathbf{x}}^{(k)}$ and $\vec{\mathbf{x}}^{(k+1)}$.

The length of the line segment between $\vec{\mathbf{x}}^{(k)}$ and $\vec{\mathbf{x}}^{(k+1)}$ can be scaled by adjusting the flexing factor ρ. As detailed in their paper, "the density-sensitive distance metric can measure the geodesic distance along the manifold, which results in any two points in the same region of high density being connected by a lot of shorter edges while any two points in different regions of high density are connected by a longer edge through a region of low density." (Wang et al., 2006) Hence, the Density Sensitive Distance Metric of Ling Wang, Liefeng Bo, and Licheng Jiao allows the distance along the path of $afedcb$ to be shorter than the path of ab (as in Figure 8) as opposed to Euclidean distance.

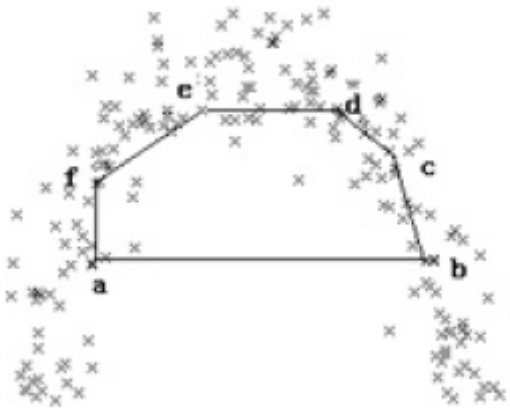


Figure 8. $\overline{af} + \overline{fe} + \overline{ed} + \overline{dc} + \overline{cb} < \overline{ab}$ (Wang et al., 2006)

Unfortunately, the Density Sensitive Distance Metric of Ling Wang, Liefeng Bo, and Licheng Jiao has a high computational cost since it assumes a complete graph over the entire dataset and then computes the shortest path between each pair of points. The Density Sensitive Distance Metric of Ling Wang, Liefeng Bo, and Licheng Jiao accomplishes everything and more that our proposed density sensitive distance measurement attempts to accomplish; however, the proposed density sensitive distance measurement will attempt to reduce the computational cost of the Density Sensitive Distance Metric of Ling Wang, Liefeng Bo, and Licheng Jiao (i.e., the cost of calculating the shortest path in the complete graph) by restricting our measure to the straight line path from the initial point $\vec{\mathbf{x}}^{(o)}$ to the final point $\vec{\mathbf{x}}^{(f)}$ while traveling over a kernel density estimation.

C. PRINCIPAL COMPONENT ANALYSIS (PCA)

Principal Component Analysis (PCA) is the linear projection that minimizes the mean squared distance between data points and their projections (Bishop, 2007). Principal Component Analysis decomposes a dataset (usually through the dataset's covariance matrix) into a set of eigenvalues and eigenvectors that represents the directions of highest to lowest variance along an orthonormal basis where the principal

eigenvector points in the direction of the highest variance and all other eigenvectors are orthogonal and point in the directions of the next highest variance.

For this work, PCA will be used to extract the maximum "lateral" variance (i.e., the maximum eigenvalue of the covariance matrix $\vec{\vec{\Sigma}}$) for each dataset in order to determine a scale γ applied to a kernel density estimate.

III. DENSITY SENSITIVE DISTANCE MEASUREMENT

A. DEFINITION

Let $\vec{\mathbf{x}}_{o\to f}\left(t\right)=\left(1-t\right)\vec{\mathbf{x}}^{(o)}+\left(t\right)\vec{\mathbf{x}}^{(f)}$ be the linear interpolation from an initial point $\vec{\mathbf{x}}^{(o)}$ (the source point) to a final point $\vec{\mathbf{x}}^{(f)}$ (the destination point) in n-dimensions and $y\left(\vec{\mathbf{x}}\right)=\gamma\left(\sum_{i=1}^{m}K_i\left(\vec{\mathbf{x}}\right)\right)$ be the scaled kernel density estimate of the dataset over which distances will be measured where $\gamma\in\mathbb{R}$ is the scale (or gain) of y, m is the number of data points in the dataset, and $K_i\left(\vec{\mathbf{x}}\right):\mathbb{R}^n\to\mathbb{R}$ is the kernel function centered at the i-th data point in that dataset (i.e., y is the sum of kernel values at $\vec{\mathbf{x}}$ where a kernel is placed at every data point in the entire dataset). Then, the density sensitive distance measurement we are proposing is

$$\text{distance}\left(\vec{\mathbf{x}}^{(o)},\vec{\mathbf{x}}^{(f)}\right)=\int_0^1\sqrt{\left(d\vec{\mathbf{x}}_{o\to f}\left(t\right)\big/dt\right)^T\left(d\vec{\mathbf{x}}_{o\to f}\left(t\right)\big/dt\right)+\left(dy\left(\vec{\mathbf{x}}_{o\to f}\left(t\right)\right)\big/dt\right)^2}\,dt$$

where $d\vec{\mathbf{x}}_{o\to f}\left(t\right)\big/dt=\vec{\mathbf{x}}^{(f)}-\vec{\mathbf{x}}^{(o)}$ is the derivative of the linear interpolation with respect to t and $dy\left(\vec{\mathbf{x}}_{o\to f}\left(t\right)\right)\big/dt$ is the derivative of $y\left(\vec{\mathbf{x}}\right)$ as it travels from the initial point $\vec{\mathbf{x}}^{(o)}$ to the final point $\vec{\mathbf{x}}^{(f)}$.

The kernel function that will be used in this work is the probability density function for the spherical multivariate Normal distribution given by:

$$K_i\left(\vec{\mathbf{x}}\right)=\frac{1}{\left(2\pi\right)^{\frac{n}{2}}\sigma^n}\exp\left(-\frac{1}{2}\sum_{j=1}^{n}\frac{\left(x_j-x_j^{(i)}\right)^2}{\sigma^2}\right)$$

where σ is the radius of the sphere which is covered by the kernel (detailed in Appendix B), $\vec{\mathbf{x}}^{(i)}$ is the i-th data point in the dataset with components $x_j^{(i)}$ for $j = 1,\ldots,n$, and n is the dimension of the dataset.

Therefore, this density sensitive distance measurement is the line integral from $\vec{\mathbf{x}}^{(o)}$ to $\vec{\mathbf{x}}^{(f)}$ as it travels along the surface of the scaled kernel density estimation of a dataset. Each dataset (or data subset) will most likely have a different kernel density estimation and the line integral from $\vec{\mathbf{x}}^{(o)}$ to $\vec{\mathbf{x}}^{(f)}$ will be sensitive to the local density of the data over which it will measure.

B. PURPOSE

The purpose of the density sensitive distance measurement is to take into account the density of a set of data when determining how similar any given point is to the dataset. If the set of data is highly concentrated, then a point that is part of that set should be at locations that mimic that concentration or else a penalty should be incurred. Similarly, if a set of data is greatly dispersed, then a point that is part of that set should also be at positions that imitate that level of dispersion or a similar price should be paid.

C. PARAMETERS

Based on the definition of this density sensitive distance measurement, there are two parameters that need to be determined before a measurement can be taken: the kernel bandwidth and the scale. For the kernel selected, the parameter that determines the kernel bandwidth is σ. The scale is determined by γ.

1. Kernel Bandwidth

There are many ways to determine the optimal kernel bandwidth when the distribution of a dataset in known or suspected. However, when the underlying distribution that generates the dataset is unknown, determining the bandwidth of the kernel becomes a matter of perspective. We are free to choose a small bandwidth to show roughness in the data. We are also free to choose a large bandwidth to show

smoothness in that same data. In other words, when we do not know what the underlying distribution is, then there is nothing for us to optimize against and we are free to choose the kernel bandwidth that best biases our results.

This is analogous to viewing a painting in a gallery. When the painter is not present to actively form the opinion of a patron by telling the patron where to stand and what to look for, then the patron must form his/her own opinion of the work. Some patrons may choose to stand close to the painting to view the detail of each brush stroke. Some patrons may stand back to view the entire work as a whole without delving into any of its detail. And some may search for a happy medium between the two. Almost all the patrons will assign meaning to some portion of the work that was unintended by the painter. However, the perspective of every patron is valid even though certain perspectives may conflict. This freedom of perspective allows patrons to see what they want to see.

Likewise, determining the kernel bandwidth of an unknown distribution is an exercise in perspective. Since we do not know if we have enough data to absolutely assert one distribution over another and since we cannot be completely certain that the available data is a true random sample from the greater population (since we do not know the greater population), the kernel bandwidth can be reduced to a matter of perspective. For our purposes, we know the kernel we are using, the probability density function of the spherical multivariate Normal distribution, has a strong additive effect when the centers of two or more of these kernels are within 2σ of each other (as in Figure 9). This additive effect causes the overall kernel density estimation to appear smooth. In this case, a kernel density estimate appears smooth when the number of extrema in the estimate are reduced to the relevant extrema, the extrema that best conform to the density we want to see. However, if too many kernel centers are within 2σ of each other, then there is too much of an additive effect and the resulting kernel density estimation is overly smooth (as in Figure 10). In this case, a kernel density estimate is overly smooth when extrema, believed to conform to the density we want to see, are eliminated by the additive effective of the kernels.

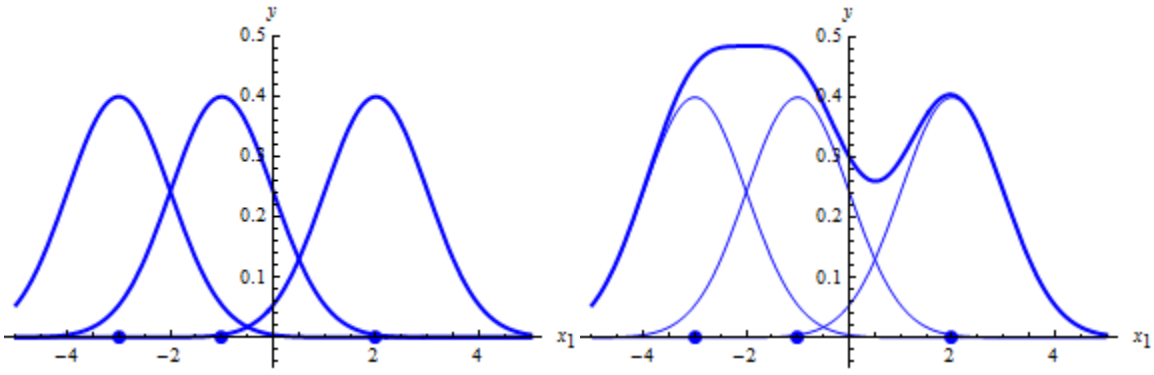


Figure 9. The additive effect that smoothes the kernel density estimation when kernel centers are within 2σ of each other. Here, $\sigma = 1$.

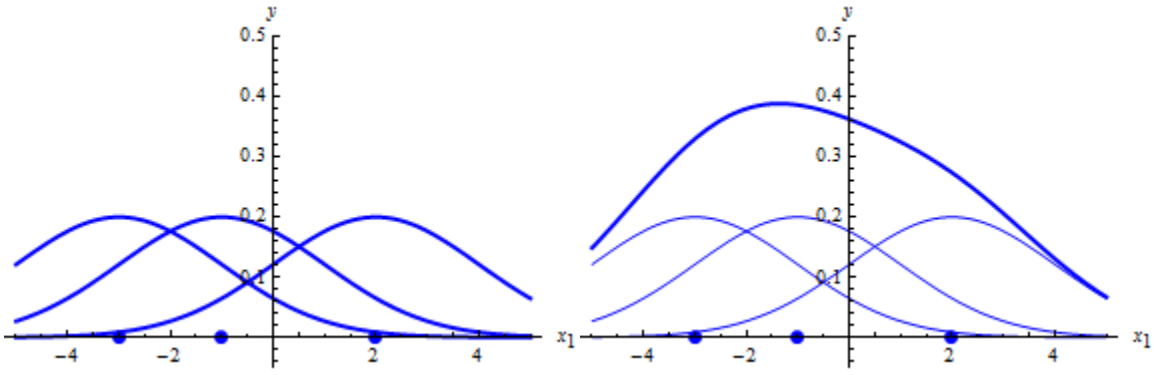


Figure 10. Over-smoothing the kernel density estimation when too many kernel centers are within 2σ of each other. Here, $\sigma = 2$.

Since we arbitrarily desire the kernel density estimation to appear smooth, but not too smooth, then we will take advantage of this additive effect and choose a σ that causes groups of centers to be within σ of each other. To avoid over-smoothing, we collect the distances of k-th nearest neighbor from each datum in the dataset, bin these distances, and choose σ that corresponds to the middle distance associated with the bin that holds the maximum number of these k-th nearest neighbor distances. If we choose k to be small compared to the size of a dataset, then our σ will be small as well and the kernel density estimation will be rougher. If we choose k to be large compared to the

size of a dataset, then our σ will be large as well and the kernel density estimation will be smoother. We found that starting with a k that is approximately 15% of the size of the dataset yields acceptable results during cross-validation when maximizing accuracy, precision, recall, or various combinations.

Since collecting the pair-wise distances can be quite expensive when the size of the dataset is high, we can treat the distances between each datum in a dataset as a population. Moreover, since we know and have access to the entire population of pair-wise distances, we can randomly sample these distances in order to come up with an acceptable kernel bandwidth. For this work, when a dataset contains over 1000 data points, we first randomly sample up to 1000 data points from that dataset, find the pair-wise distances between those randomly sampled points, and complete the previously described to-avoid-over-smoothing routine above on this random sample.

Note: As $\sigma \to \infty$, the kernel density estimate approaches a flat plane and this density sensitive distance measurement approaches Euclidean distance.

2. Scale

As with kernel bandwidth, there are many ways to determine scale. Scale γ effects the amplitude (or gain) of the kernel density estimation for a dataset (as in Figure 11). If $\gamma > 1$, then the kernel density estimate will be amplified (i.e., the gain will be turned up) and the existing dataset will produce y-values with a higher variance (as in Figure 12). If $0 < \gamma < 1$, then the amplitude of kernel density estimate will be reduced (i.e., the gain will be turned down) and the existing dataset will produce y-values with less variance (as in Figure 13).

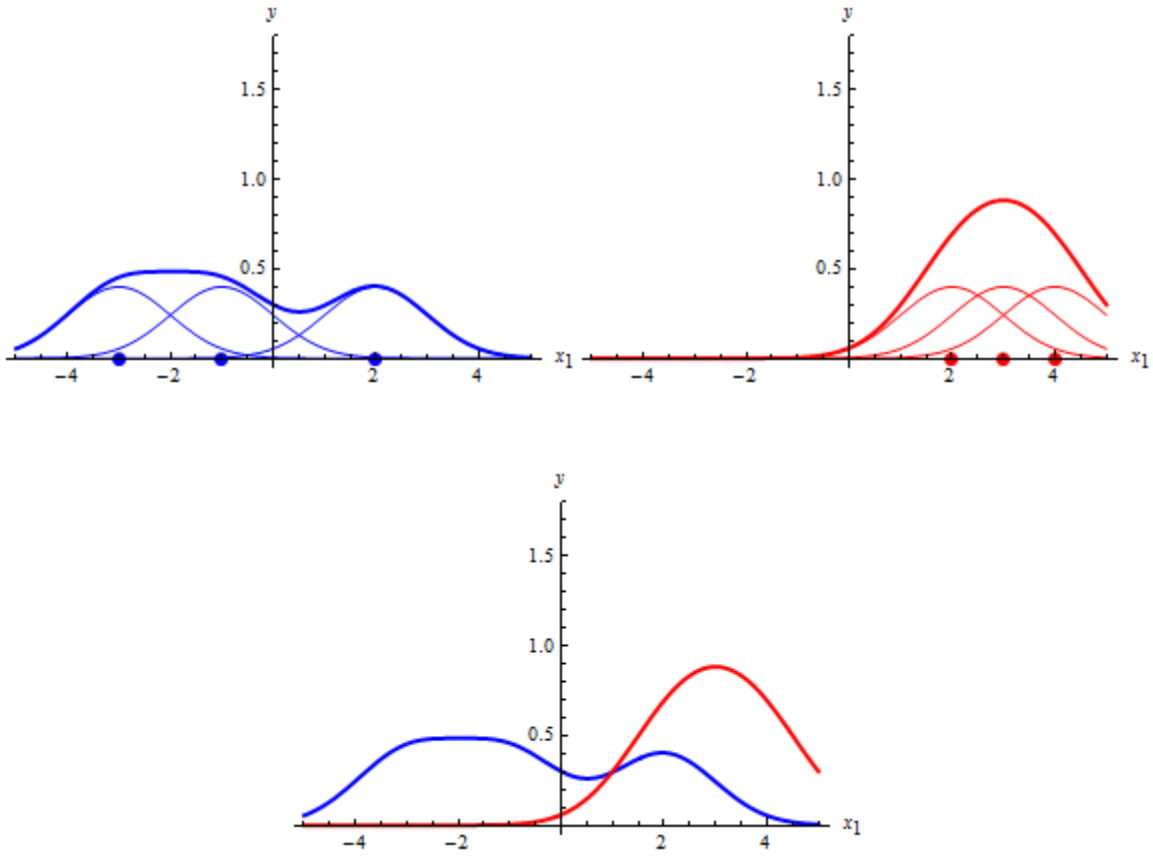


Figure 11. The kernel density estimation with $\gamma = 1$ for two separate datasets.

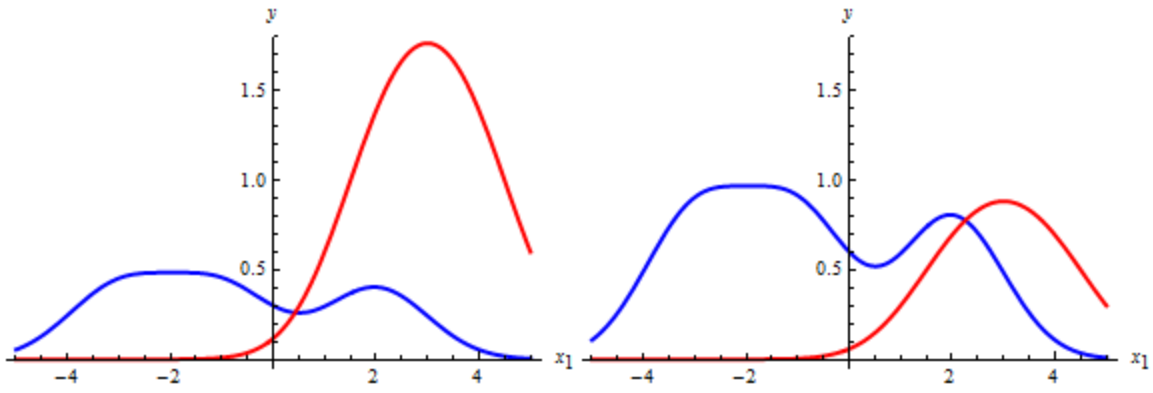


Figure 12. Doubling the scale associated with a dataset. Left, $\gamma = 1$ for the blue kernel density estimate and $\gamma = 2$ for the red kernel density estimate. Right, $\gamma = 2$ for the blue kernel density estimate and $\gamma = 1$ for the red kernel density estimate.

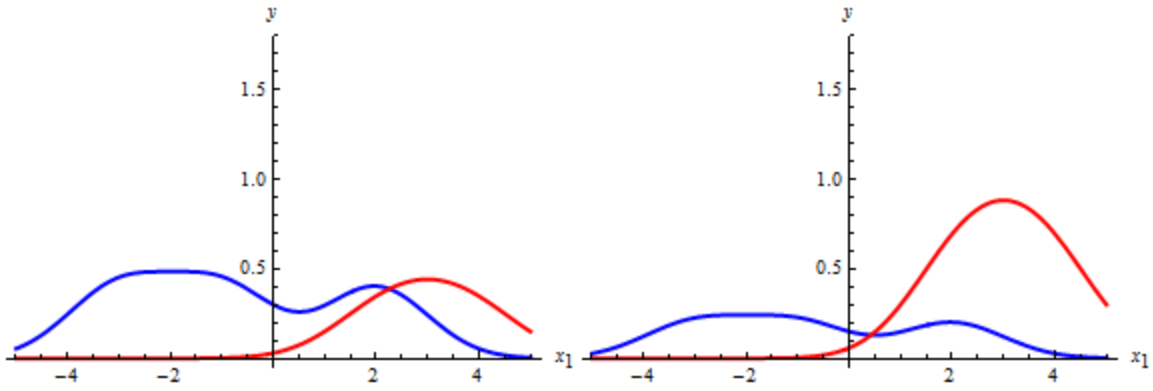


Figure 13. Halving the scale associated with a dataset. Left, $\gamma = 1$ for the blue kernel density estimate and $\gamma = 1/2$ for the red kernel density estimate. Right, $\gamma = 1/2$ for the blue kernel density estimate and $\gamma = 1$ for the red kernel density estimate.

Moreover, γ can be used to change the variance of the y-values of a dataset. Since we know that variance can be estimated by the following equation:

$$\begin{aligned}\sigma^2_{y\text{-values}} &= \frac{1}{m-1}\sum_{i=1}^{m}\left(y_i - \overline{y}\right)^2 \\ &= \frac{1}{m-1}\sum_{i=1}^{m}\left(y_i - \frac{1}{m}\sum_{j=1}^{m} y_j\right)^2\end{aligned}$$

then γ can change the variance estimate of the y-values by the following:

$$
\begin{aligned}
\sigma^2_{\substack{\text{amplified}\\ y\text{-values}}} &= \frac{1}{m-1}\sum_{i=1}^{m}\left(\gamma y_i - \frac{1}{m}\sum_{j=1}^{m}\gamma y_j\right)^2 \\
&= \frac{1}{m-1}\sum_{i=1}^{m}\left(\gamma y_i - \gamma\frac{1}{m}\sum_{j=1}^{m} y_j\right)^2 \\
&= \frac{1}{m-1}\sum_{i=1}^{m}\left(\gamma\left(y_i - \frac{1}{m}\sum_{j=1}^{m} y_j\right)\right)^2 \\
&= \frac{1}{m-1}\sum_{i=1}^{m}\gamma^2\left(y_i - \frac{1}{m}\sum_{j=1}^{m} y_j\right)^2 \\
&= \gamma^2\left(\frac{1}{m-1}\sum_{i=1}^{m}\left(y_i - \frac{1}{m}\sum_{j=1}^{m} y_j\right)^2\right) \\
&= \gamma^2\left(\sigma^2_{y\text{-values}}\right)
\end{aligned}
$$

Hence, $\gamma = \sqrt{\sigma^2_{\substack{\text{amplified}\\ y\text{-values}}} \Big/ \sigma^2_{y\text{-values}}}$ which implies that to change the variance of the y-values produced by the kernel density estimate for a dataset, we only need to multiply the kernel density estimate by the following:

$$
\gamma = \sqrt{\frac{\sigma^2_{\text{desired } y \text{ variance}}}{\sigma^2_{\text{current } y \text{ variance}}}}
$$

Note: That if $\gamma = 0$, then this density sensitive distance measurement is identical to Euclidean distance.

D. IMPLEMENTATION

The line integral for this density sensitive distance measurement can be implemented using at least two methods: local adaptive quadrature on the integrand (Burden & Faires, 2005) or local adaptive Euclidean distance on the scaled kernel density estimation. Local adaptive quadrature can be faster; however, the derivative of $y\left(\vec{\mathbf{x}}\right)$ must be calculated. If that is not desirable or even possible, then we can use local adaptive Euclidean distance directly on the scaled kernel density estimation (as in Figure 14). For local adaptive Euclidean distance, we simply choose various points along the

path from $\vec{\mathbf{x}}^{(o)}$ to $\vec{\mathbf{x}}^{(f)}$, calculate their respective scaled kernel density estimations, and measure the Euclidean distance from point to point. Then we divide each implied line segment in two and take the Euclidean distance of those two new segments. We iterate until the change in distance between the whole segment and the two half segments is less than an established threshold. When the distances of all the segments have been calculated, then the sum of all those segment distances approximates the line integral distance.

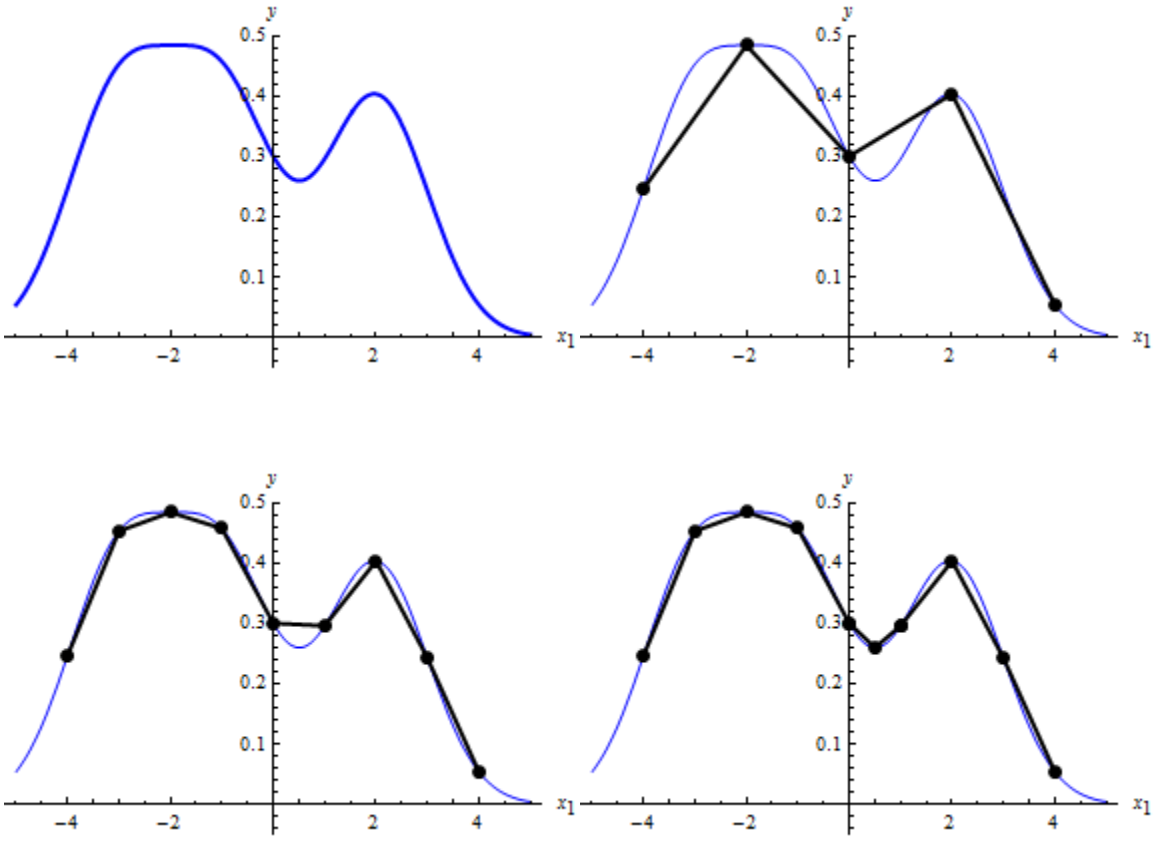


Figure 14. Successive iterations of local adaptive Euclidean distance on the scaled kernel density estimate from -4 to 4 in order to approximate the line integral from -4 to 4.

E. STRENGTHS

The main strength of this density sensitive distance measurement is that it takes into account the density of the dataset over which it measures. In so doing, the density sensitive distance measurement provides a more shape-conforming distance for classification than Euclidean distance alone. For instance, if we only look to the immediate nearest neighbor using Euclidean distance for classification, then this is equivalent to the classifying a point based on its location in the Voronoi diagram. For the circular datasets used in the Introduction, we would have a star shaped pattern for classification, vice a circular one (as in Figure 15 and Figure 16). However, if we let $v_{\substack{\text{blue } y \\ \text{variance}}}$ be the y-value variance for the blue class, $v_{\substack{\text{red } y \\ \text{variance}}}$ be the y-value variance for the red class, and $v_{\substack{\text{blue max} \\ \text{lateral variance}}}$ and $v_{\substack{\text{red max} \\ \text{lateral variance}}}$ be the maximum lateral variances for the blue and red classes, respectively, then we can look to the immediate nearest neighbor using the proposed density sensitive distance for classification and achieve much more shape-conforming results (as in Figure 17 and Figure 18).

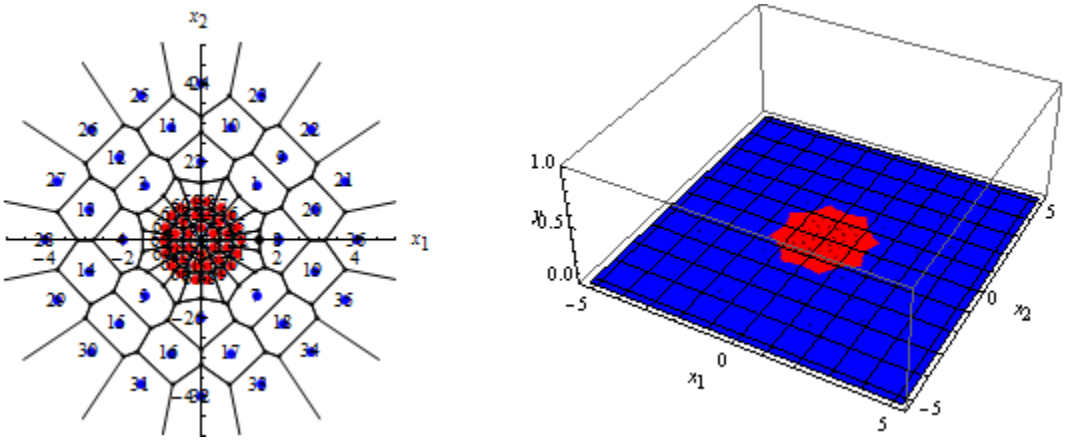


Figure 15. The Voronoi diagram and 1-nearest neighbor classification using Euclidean distance on the datasets from the Introduction.

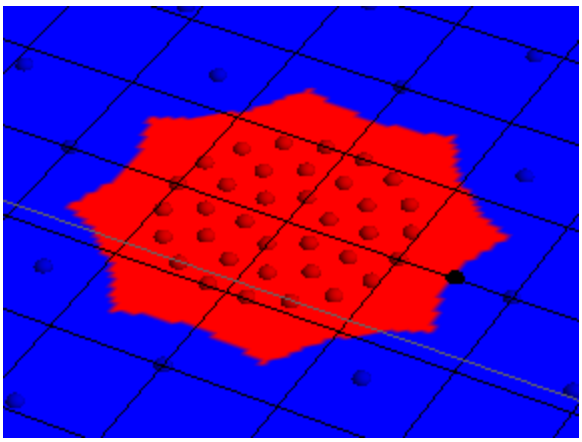


Figure 16. Close up of 1-nearest neighbor classification using Euclidean distance on the datasets from the Introduction.

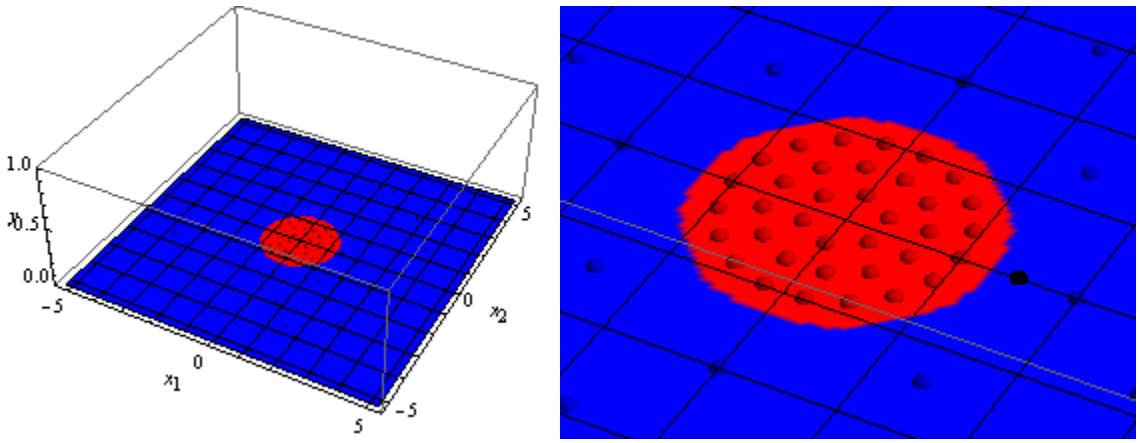


Figure 17. 1-nearest neighbor classification using density sensitive distance with

$$\sigma_{\text{blue}} = \sqrt{v_{\substack{\text{blue max} \\ \text{lateral variance}}}}, \quad \sigma_{\text{red}} = \sqrt{v_{\substack{\text{red max} \\ \text{lateral variance}}}},$$

$$\gamma_{\text{blue}} = \sqrt{\min\left\{v_{\substack{\text{blue } y \\ \text{variance}}}, v_{\substack{\text{red } y \\ \text{variance}}}\right\} \Big/ v_{\substack{\text{blue } y \\ \text{variance}}}} \text{ and}$$

$$\gamma_{\text{red}} = \sqrt{\min\left\{v_{\substack{\text{blue } y \\ \text{variance}}}, v_{\substack{\text{red } y \\ \text{variance}}}\right\} \Big/ v_{\substack{\text{red } y \\ \text{variance}}}} \text{ on the datasets from the Introduction.}$$

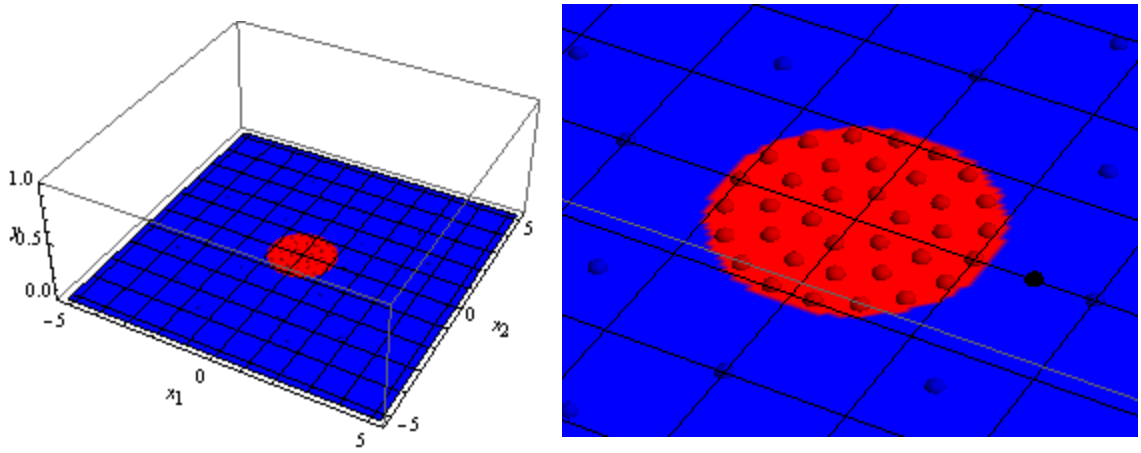


Figure 18. 1-nearest neighbor classification using density sensitive distance with $\sigma_{\text{blue}} = \sqrt{v_{\substack{\text{blue max} \\ \text{lateral variance}}}}$, $\sigma_{\text{red}} = \sqrt{v_{\substack{\text{red max} \\ \text{lateral variance}}}}$, $\gamma_{\text{blue}} = \sqrt{\max\left\{v_{\substack{\text{blue } y \\ \text{variance}}}, v_{\substack{\text{red } y \\ \text{variance}}}\right\} \Big/ v_{\substack{\text{blue } y \\ \text{variance}}}}$ and $\gamma_{\text{red}} = \sqrt{\max\left\{v_{\substack{\text{blue } y \\ \text{variance}}}, v_{\substack{\text{red } y \\ \text{variance}}}\right\} \Big/ v_{\substack{\text{red } y \\ \text{variance}}}}$ on the datasets from the Introduction.

F. WEAKNESSES

The density sensitive distance measurement is not a distance metric. In order for a measurement to be considered a metric, the triangular inequality must hold (Zezula, 2006). In other words, to be a metric, the following property must hold:

$$\forall x, y, z \in S, \text{distance}\left(x, z\right) \leq \text{distance}\left(x, y\right) + \text{distance}\left(y, z\right)$$

The triangular inequality does not hold for this density sensitive distance measurement. For instance, given $\left(-4,-4\right)$, $\left(4,-4\right)$, and $\left(4,4\right)$ and the scaled kernel density estimate in Figure 19, we would have the following:

$$\text{distance}\left(\left(-4,-4\right),\left(4,4\right)\right) > \text{distance}\left(\left(-4,-4\right),\left(4,-4\right)\right) + \text{distance}\left(\left(4,-4\right),\left(4,4\right)\right)$$

for density sensitive distance. Hence, the triangle inequality does not hold for this density sensitive distance measure. Therefore, the density sensitive distance measurement is a measurement, not a metric.

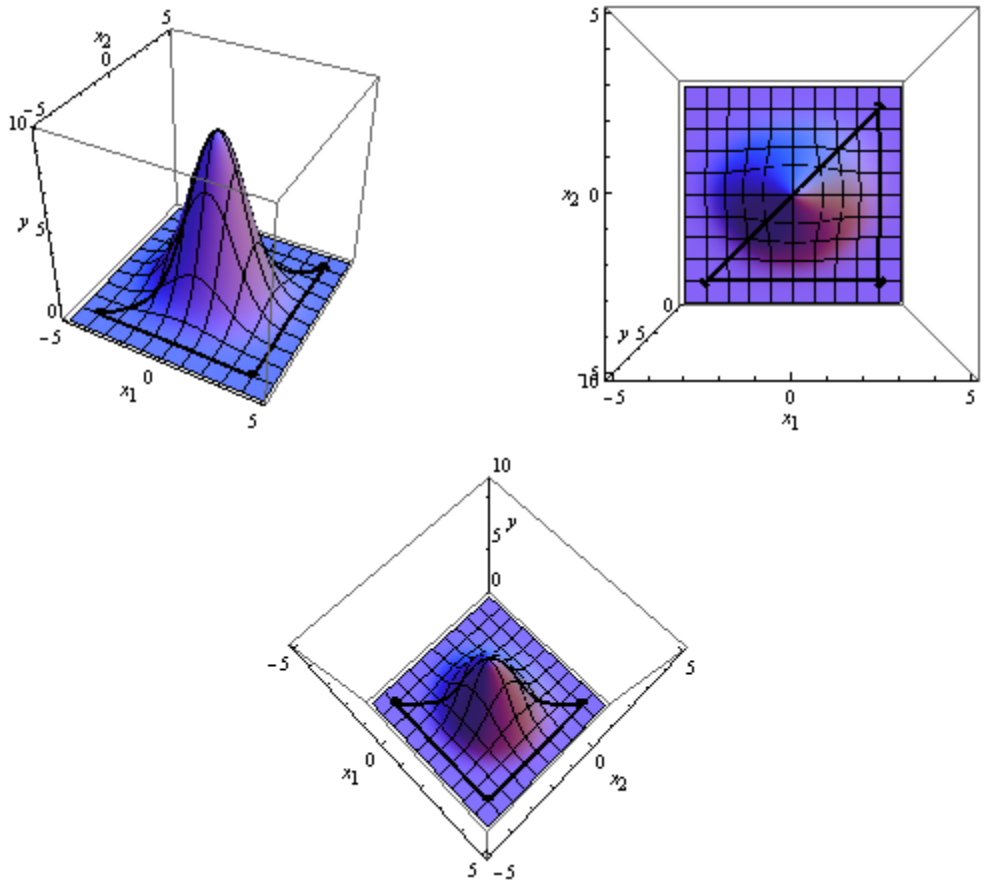


Figure 19. Different perspective views on the same scaled kernel density estimation that demonstrate that the triangle inequality does not hold for this density sensitive distance measurement.

Additionally, if we implement this density sensitive distance measurement using either local adaptive quadrature or local adaptive Euclidean distance, then we need to be conscious of the fact that a poor choice in where a line segment is broken can lead to incorrect results when calculating the line integral (as in Figure 20). If a line segment is broken and the difference between the length of the original line and the lengths of the resulting two line segments is under a threshold, then locally adaptive routines assume they have adapted to their goal with a specified tolerance. If they have not properly adapted, then the locally adaptive routine will return incorrect results for the line integral.

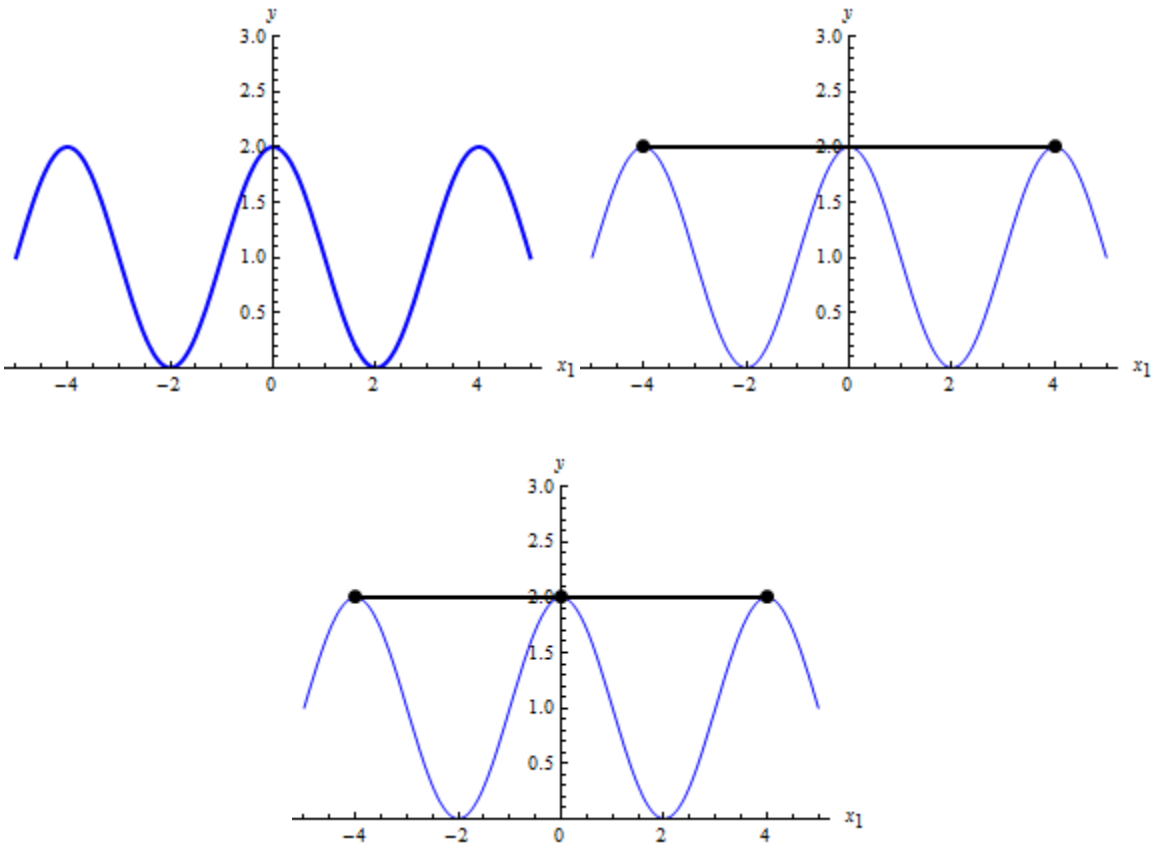


Figure 20. A poor choice for the break in a line segment that will stop further locally adaptive line segments from being generated in the computation of the line integral from -4 to 4.

Also, even though improvements have been made in shape-conforming classification, the potential exists to get odd classification results from certain combinations of kernel bandwidth and scale (as in Figure 21). In Figure 21, if a datum falls in line with regions that we would usually consider to blue or red, then everything is as expected; however, if a datum is an outlier and falls far enough out of the traditional boundaries, then the datum is classified as red, even though blue is closer from a Euclidean distance point of view.

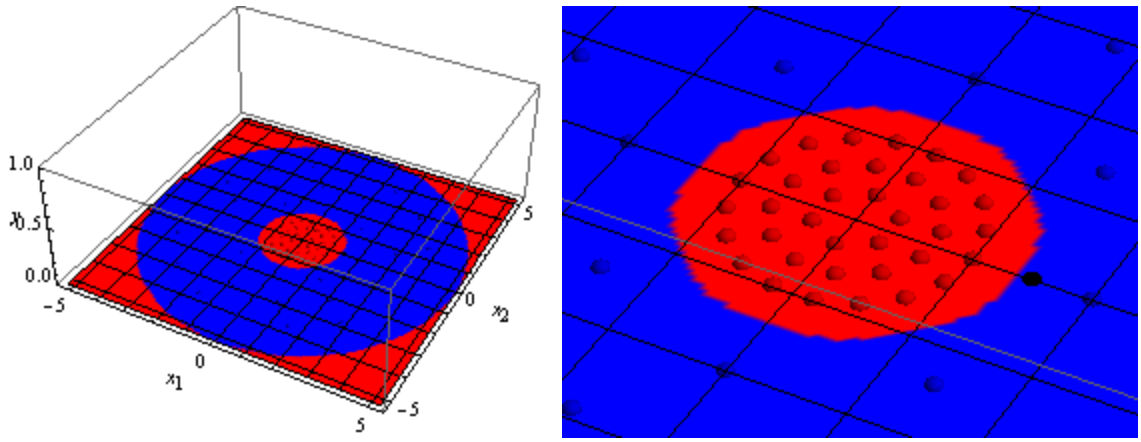


Figure 21. 1-nearest neighbor classification using density sensitive distance with $\sigma_{\text{blue}} = \sqrt{v_{\substack{\text{blue max} \\ \text{lateral variance}}}}$, $\sigma_{\text{red}} = \sqrt{v_{\substack{\text{red max} \\ \text{lateral variance}}}}$, $\gamma_{\text{blue}} = \sqrt{v_{\substack{\text{blue max} \\ \text{lateral variance}}} \Big/ v_{\substack{\text{blue } y \\ \text{variance}}}}$ and $\gamma_{\text{red}} = \sqrt{v_{\substack{\text{red max} \\ \text{lateral variance}}} \Big/ v_{\substack{\text{red } y \\ \text{variance}}}}$ on the artificial dataset.

Note that as odd as the results of Figure 21 are, we will retain its choice of γ for classification later in this work.

Lastly, since the kernel density estimate used by the proposed density sensitive distance measurement is non-parametric, the entire dataset must be retained and repeatedly iterated over for each measurement, not just representative samples from that dataset. Therefore, there is a high computational cost to the proposed density sensitive distance measurement.

THIS PAGE INTENTIONALLY LEFT BLANK

IV. METHODOLOGY

To test the utility of the proposed density sensitive distance measurement, we use this distance measurement to perform supervised learning. For k-Nearest Neighbor classification using the density sensitive distance measurement, we train and test on two real-world datasets. We compare this to k-Nearest Neighbor classification using Euclidean distance. Lastly, to put both of these results into context, we perform Support Vector Machine and Random Forests classification to see how well classification using this density sensitive distant measurement stands up against modern supervised learning algorithms. For all classifiers, we use stratified 10-fold cross validation to obtain the generalization error of each classifier, record the overall accuracy and error rate, and document the precision and recall for each class.

A. DATASETS

We test the proposed density sensitive distance measurement on two datasets - the entire Wisconsin Diagnostic Breast Cancer (WDBC) dataset and a portion of the MNIST Database of Handwritten Digits.

1. The Wisconsin Diagnostic Breast Cancer (WDBC) Dataset

The Wisconsin Diagnostic Breast Cancer (WDBC) dataset is from the University of California, Irvine, repository (*UCI Machine Learning Repository: Breast Cancer Wisconsin (Diagnostic) Data Set.*). This is a small multivariate dataset with 569 total datum where each datum consists of 30 real-valued components. Each datum is constructed from a digitized image of a fine needle aspirate (FNA) of a breast mass. The datum represent characteristics of the cell nuclei present in the image. Ten real-valued components are computed for each cell nucleus:

1) radius (mean of distances from center to points on the perimeter),

2) texture (standard deviation of gray-scale values),

3) perimeter,

4) area,

5) smoothness (local variation in radius lengths),

6) compactness ($\mathrm{perimeter}^2 \big/ \mathrm{area} - 1$),

7) concavity (severity of concave portions of the contour),

8) concave points (number of concave portions of the contour),

9) symmetry, and

10) fractal dimension ($\text{"coastline approximation"} - 1$).

The mean, standard error, and "worst" or largest (mean of the three largest values) of these components were computed for each image, resulting in 30 total components. For instance, the first component is the mean of the radius, the 11th component is the standard error of the radius, and 21 component is the worst radius.

Of the 569 total datum, 357 datum represent benign tumors and 212 represent malignant ones.

2. The MNIST Database of Handwritten Digits

The MNIST Database of Handwritten Digits is a subset of a larger database available from the National Institute of Standards and Technology (NIST) (*MNIST Handwritten Digit Database, Yann LeCun and Corinna Cortes.*). The MNIST database was constructed from NIST's Special Database 1 and Special Database 3 which contain binary images of handwritten digits. For the MNIST database, the original binary images from the NIST databases were size normalized to fit in 20×20 pixel windows while preserving their aspect ratio. The resulting images contain grey levels as a result of the anti-aliasing technique used by the normalization algorithm. Each 20×20 pixel image was centered in a 28×28 pixel window by computing the center of mass of the pixels, and translating the image so as to position this point at the center of this 28×28 pixel window.

The MNIST database consists of two sets of images - a training set of 60,000 images and a testing set of 10,000 images. The 60,000 pattern training set contains examples from approximately 250 different writers.

For this work, we only use the handwritten ones, twos, and threes from the MNIST training set. This is a medium sized multivariate dataset with 18,831 total datum where each datum consists of 784 integer-valued components with integers ranging from 0 to 255. Of the 18,831 total datum, 6,742 datum represent handwritten ones, 5,958 datum represent handwritten twos, and 6,131 datum represent handwritten threes. Examples of the handwritten ones, twos, and threes are shown in original and enlarged sizes in Figure 22, Figure 23, and Figure 24.

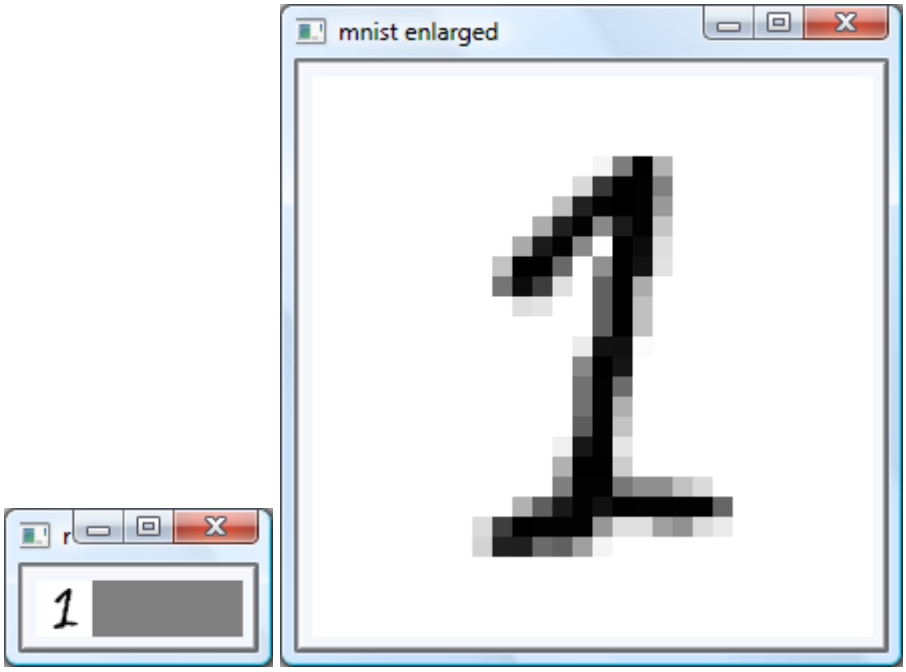


Figure 22. An example of a handwritten one from the MNIST training dataset. Left, the original size of the example. Right, the enlarged size.

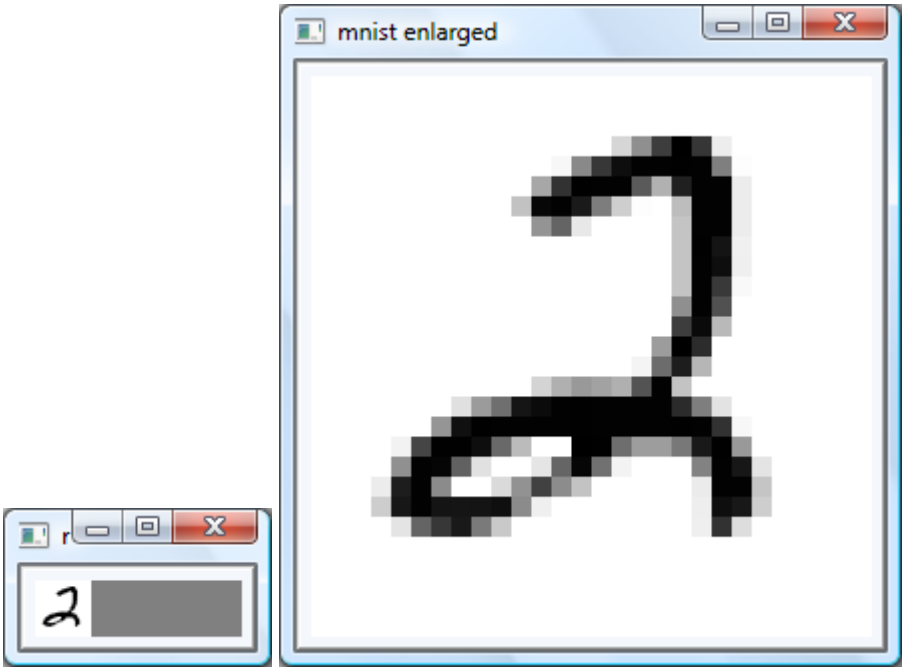


Figure 23. An example of a handwritten two from the MNIST training dataset. Left, the original size of the example. Right, the enlarged size.

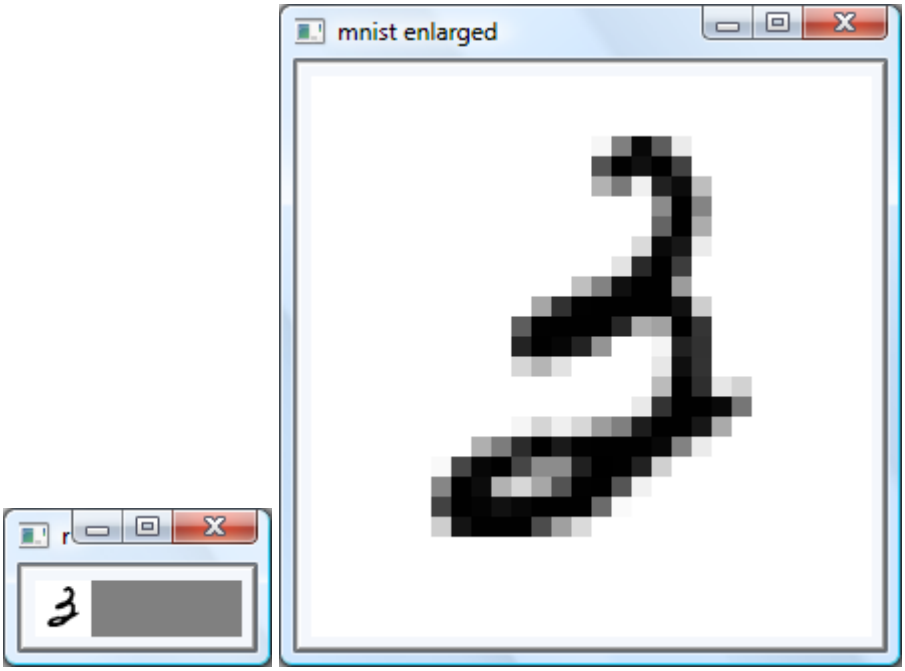


Figure 24. An example of a handwritten three from the MNIST training dataset. Left, the original size of the example. Right, the enlarged size.

B. SUPERVISED LEARNING

Supervised learning is learning in which an algorithm receives a set of input datum and their corresponding output datum (i.e., a training dataset), trains on this data to find a function of the input data that approximates the known output data, and then uses that trained function on unknown input data (i.e., on a testing dataset) (Izenman, 2008). The input data may contain continuous or categorical values. For classification, the output data consists of categorical values, usually called labels. The goal of the learning algorithm for classification is to minimize the error incurred during the testing phase while only training on the training dataset. In essence, supervised learning is analogous to classroom instruction. A teacher presents each student with a set of various problems and their respective correct answers, the student then conceptualizes those problems and their respective answers, and finally, the student is tested on previously unseen problems and a grade is recorded.

C. CLASSIFICATION ALGORITHMS

For this work, we use the following classification algorithms: k-Nearest Neighbor classification using the proposed density sensitive distance measurement, k-Nearest Neighbor classification using Euclidean distance, Support Vector Machine, and Random Forests.

1. k-Nearest Neighbor Classification

k-Nearest Neighbor classification is classification of a testing datum based on the majority vote of the class labels of k most similar training data. For this work, similarity will be determined by our density sensitive distance measurement and by Euclidean distance.

Although 1-Nearest Neighbor is a sub-optimal procedure, its error rate can be bounded from below. Given an unlimited amount of training data, 1-Nearest Neighbor classification has an error rate guaranteed to be no worse than twice the Bayes error rate, the minimum possible error rate (Duda, Hart, & Stork, 2000).

2. Support Vector Machine Classification

Support Vector Machine classification projects training data into a higher-dimensional space by creating new dimensions from combinations of the original dimensions and then finds the hyperplane that best separates the classes of data in that higher-dimension (Bradski & Kaehler, 2008). Kernel functions, such as the polynomial kernel or the radial basis function kernel, are used to creating those new dimensions from combinations of original dimensions. During testing, incoming data are projected into the higher-dimension using the kernel function, an inner-product is taken based on the normal vector of the class-separating hyperplane, and the sign of that inner product determines the classification of the data. For multiple-class classification problems, multiple hyperplanes can be constructed. For example, in a three class classification problem, the first hyperplane can separate class 1 data from non-class 1 data (i.e., class 2 & 3 data). The second hyperplane can separate class 2 data from non-class 2 data (i.e., class 3 data). During the testing phase, a datum that is initially classified as a non-class 1 datum need only look to the second separating hyperplane to determine if that datum should be classified as class 2 or class 3.

For this work, we use the Support Vector Machine implementation in OpenCV 1.1 (OpenCV 1.1 2008).

3. Random Forests Classification

Random Forests (*Random Forests* 2009) classification randomly constructs multiple decision trees based on training data. During testing, each tree votes on the classification of incoming datum. Incoming datum are classified based on the class with the most votes. Below we give a brief description of Random Forests. For a more detailed explanation, refer to (*Random Forests* 2009).

In Random Forests, each tree is constructed as follows:

1) If the size of the training set is m, then sample m cases at random with replacement and grow the tree from this sample set.

2) If there are n components, a number $k << n$ is specified such that at each node, k of n components are selected at random and the best split on these k is used to split the node. The value of k is held constant during construction of the entire forest.

3) No tree is pruned and each tree is grown to the largest extent possible.

The Random Forest error rate depends on two things: 1) the correlation between any two trees in the forest and 2) the strength of each individual tree in the forest. Increasing the correlation between trees increases the error rate. Increasing the strength of the individual trees decreases the error rate. Reducing k reduces both the correlation and the strength. Increasing k increases correlation and strength. Hence, we need to find the k that optimizes these parameters.

For this work, we use the Random Forests implementation in OpenCV 1.1 (OpenCV 1.1 2008).

D. STRATIFIED 10-FOLD CROSS VALIDATION

For this work, the classification algorithms are trained and tested using stratified 10-fold cross validation.

For c different classes, we separate the original data set into c class data sets where each class data set only contains data with the same class label; in other words, the data in each of these class data sets are from the same class.

Over 10 iterations, we then separate each class data set into two different sets - a class training set and a class testing set. For the first iteration, the first 10% of each class data set is used for testing and the remaining 90% is used for training. For the second iteration, the next 10% of each class data set is used for testing and the remaining 90% is used for training. The data in third through the tenth iteration is divided similarly so that all of the data in the class data sets are used for testing during exactly one of the iterations.

Cross validation is used to estimate prediction error (Hastie, Tibshirani, & Friedman, 2001). Cross validation directly estimates the generalization error when a classification algorithm is applied to an independent sample testing dataset.

E. STATISTICS

For each of the 10 folds of a cross validation, we record the confusion matrix for that fold. From that confusion matrix, we determine the overall accuracy and error rate for that fold. Additionally, the confusion matrix is also used to determine the precision and recall of each class during that fold. At the end of the 10 folds of the cross validation, we find the mean and the standard deviation of overall accuracy and error. We also find the mean and standard deviation of the precision and recall of each class.

1. Confusion Matrix

The confusion matrix is a matrix that consists of rows that represent predicted classes and columns that represent actual classes from the testing phase of cross validation (as in Table 1.)

		Actual	
		Class of Interest	**All Other Classes**
Predicted	**Class of Interest**	True Positive (TP)	False Positive (FP)
	All Other Classes	False Negative (FN)	True Negative (TN)

Table 1. The Confusion Matrix.

During the testing phase of cross validation, a classification algorithm will predict the class of a testing datum. Since we also know the actual class of the testing datum during cross validation, then we can increment the cell in the confusion matrix that corresponds to the class the classifier predicted for the testing datum (i.e., the row) and the actual class of the testing datum (i.e., the column).

When determining true positives, false positives, false negatives, and true negatives, we must first select a class of interest. For instance, if we have the confusion matrix show in Table 2. and we choose Class 1 as our class of interest, then the count of

our True Positives (i.e., Predicted: Class 1 and Actual: Class 1) would be 10, the count of our False Positives (i.e., Predicted: Class 1 and Actual: All Other Classes) would be $1+2=3$, the count of our False Negatives (i.e., Predicted: All Other Classes and Actual: Class 1) would be $3+5=8$, and the count of our True Negatives (i.e., Predicted: All Other Classes and Actual: All Other Classes) would be $20+4+6+30=60$.

		Actual		
		Class 1	**Class 2**	**Class 3**
Predicted	**Class 1**	10	1	2
	Class 2	3	20	4
	Class 3	5	6	30

Table 2. An example three-class confusion matrix.

2. Overall Accuracy

Overall accuracy represents how well a classifier performed during a fold of cross validation procedure. From the confusion matrix, overall accuracy is computed by dividing the summation of the value on the main diagonal by the summation of every value in the matrix. In other words, the overall accuracy is the following:

$$\frac{\sum_{i=j} CM_{i,j}}{\sum_{i=1}^{c}\sum_{j=1}^{c} CM_{i,j}}$$

where $CM_{i,j}$ represents the i-th row and j-th column of the confusion matrix and c is the number of separate classes.

3. Overall Error Rate

Overall error rate represents how poorly a classifier performed during a fold of the cross validation procedure. Overall error rate is the opposite of overall accuracy;

however, we can also compute the overall error rate from the confusion matrix by dividing the summation of all values off of the main diagonal by the summation of every value in the matrix. In other words, the overall error rate is the following:

$$\frac{\sum_{i \neq j} CM_{i,j}}{\sum_{i=1}^{c} \sum_{j=1}^{c} CM_{i,j}}.$$

4. Precision

To calculate precision, we must first fix a class of interest. Once a class of interest is chosen, then precision can be calculated by the following:

$$\text{Precision} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Positives}}.$$

5. Recall

To calculate recall, we must first fix a class of interest. Once a class of interest is chosen, then recall can be calculated by the following:

$$\text{Recall} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Negatives}}$$

A good classification algorithm may produce high values for both precision and for recall. A poor classifier will produce high values for either precision or for recall, but not for both.

V. RESULTS

The results for classification over the Wisconsin Diagnostic Breast Cancer (WDBC) dataset and the ones, twos, and threes from the MNIST Database of Handwritten Digits using various classifiers are presented. We report the results of the two best parameterizations for each type of classifier (as determined by the overall accuracy).

A. THE WISCONSIN DIAGNOSTIC BREAST CANCER (WDBC) DATASET

1. Overall Accuracy and Error Rate

Classifier	Overall Accuracy	Overall Error Rate
k-Nearest Neighbor using the Density Sensitive Distance Measurement with $k_{\text{classification}} = 8$ and $k_{\substack{\text{kernel density} \\ \text{estimation}}} = 100$	0.943759 ± 0.0111921	0.0562408 ± 0.0111921
k-Nearest Neighbor using the Density Sensitive Distance Measurement with $k_{\text{classification}} = 10$ and $k_{\substack{\text{kernel density} \\ \text{estimation}}} = 106$	0.938525 ± 0.0146528	0.0614748 ± 0.0146528

k -Nearest Neighbor using Euclidean Distance with $k = 8$	0.938558 ± 0.0188182	0.0614424 ± 0.0188182
k -Nearest Neighbor using Euclidean Distance with $k = 10$	0.935048 ± 0.0185173	0.0649523 ± 0.0185173
Support Vector Machine using the Polynomial Kernel with $\text{Degree} = 3$, $\gamma = 10^{-6}$, $\nu = 0.1$, and $\text{coef0} = 240$	0.962938 ± 0.0348735	0.0370625 ± 0.0348735
Support Vector Machine using the Polynomial Kernel with $\text{Degree} = 5$, $\gamma = 10^{-6}$, $\nu = 0.1$, and $\text{coef0} = 2.4$	0.961275 ± 0.0233969	0.038725 ± 0.0233969
Random Forests with $\text{trees} = 20$, $\text{depth} = 20$, and $\text{split size} = 11$	0.966569 ± 0.0154972	0.0334306 ± 0.0154972
Random Forests with $\text{trees} = 20$, $\text{depth} = 20$, and $\text{split size} = 6$	0.966538 ± 0.0255376	0.0334619 ± 0.0255376

Table 3. Overall Accuracy and Error Rate for the Wisconsin Diagnostic Breast Cancer (WDBC) Dataset

2. Class Precision and Recall

	Class: Malignant		Class: Benign	
Classifier	**Precision**	**Recall**	**Precision**	**Recall**
k -Nearest Neighbor using the Density Sensitive Distance Measurement with $k_{\text{classification}} = 8$ and $k_{\substack{\text{kernel density} \\ \text{estimation}}} = 100$	0.948574 ± 0.0397934	0.901082 ± 0.0565543	0.9443 ± 0.0301339	0.969127 ± 0.0246911
k -Nearest Neighbor using the Density Sensitive Distance Measurement with $k_{\text{classification}} = 10$ and $k_{\substack{\text{kernel density} \\ \text{estimation}}} = 106$	0.952244 ± 0.037813	0.882468 ± 0.0625009	0.934446 ± 0.0321631	0.971905 ± 0.0231181
k -Nearest Neighbor using Euclidean Distance with $k = 8$	0.935527 ± 0.0488191	0.901082 ± 0.0648552	0.944229 ± 0.0353156	0.960794 ± 0.0300639
k -Nearest Neighbor using Euclidean Distance with $k = 10$	0.934084 ± 0.0381952	0.891775 ± 0.0737203	0.939503 ± 0.0398287	0.960714 ± 0.0236421

Support Vector Machine using the Polynomial Kernel with Degree $= 3$, $\gamma = 10^{-6}$, $\nu = 0.1$, and coef0 $= 240$	0.953864 ± 0.055712	0.947835 ± 0.0524313	0.969643 ± 0.0303464	0.971905 ± 0.035075
Support Vector Machine using the Polynomial Kernel with Degree $= 5$, $\gamma = 10^{-6}$, $\nu = 0.1$, and coef0 $= 2.4$	0.954192 ± 0.0468176	0.943506 ± 0.0434155	0.967224 ± 0.0248127	0.971905 ± 0.0297879
Random Forests with trees $= 20$, depth $= 20$, and split size $= 11$	0.963834 ± 0.0417267	0.948052 ± 0.0345719	0.969932 ± 0.0194968	0.977619 ± 0.0257602
Random Forests with trees $= 20$, depth $= 20$, and split size $= 6$	0.959844 ± 0.0521776	0.952814 ± 0.0383116	0.972399 ± 0.0219143	0.974762 ± 0.0337253

Table 4. Precision and Recall for each class in the Wisconsin Diagnostic Breast Cancer (WDBC) Dataset

3. Precision and Recall Curves

The following plots show precision versus recall for all 10-fold cross validation runs of all classifiers. In these curves, the scale for the Precision and Recall axes range from 0.60 to 1.00, vice 0.00 to 1.00, to emphasize the results. Note that some individual runs of stratified 10-fold cross validation of Support Vector Machine and Random Forests classification were perfect.

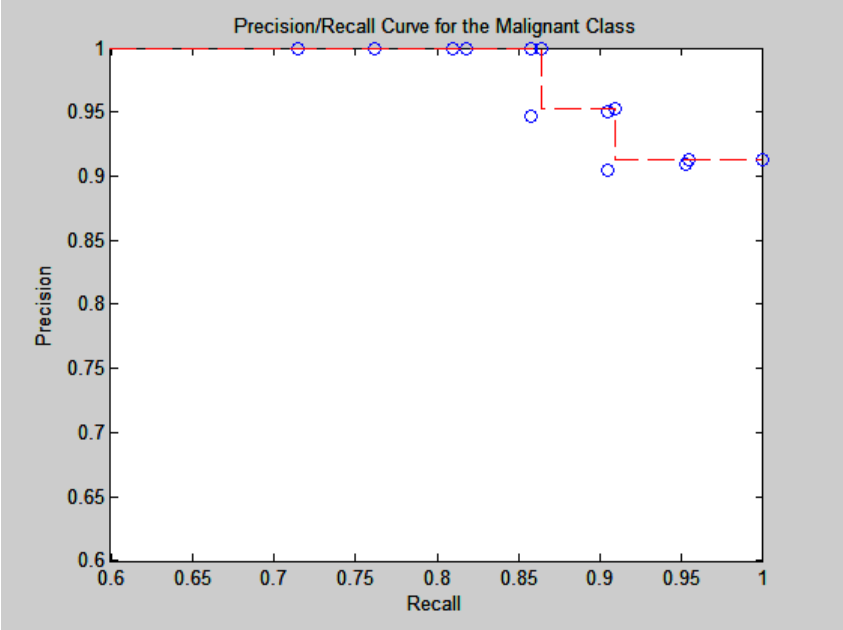


Figure 25. The Precision and Recall Curve for the k-Nearest Neighbor classifiers using the Density Sensitive Distance Measurement for the Malignant Class in the Wisconsin Diagnostic Breast Cancer (WDBC) Dataset

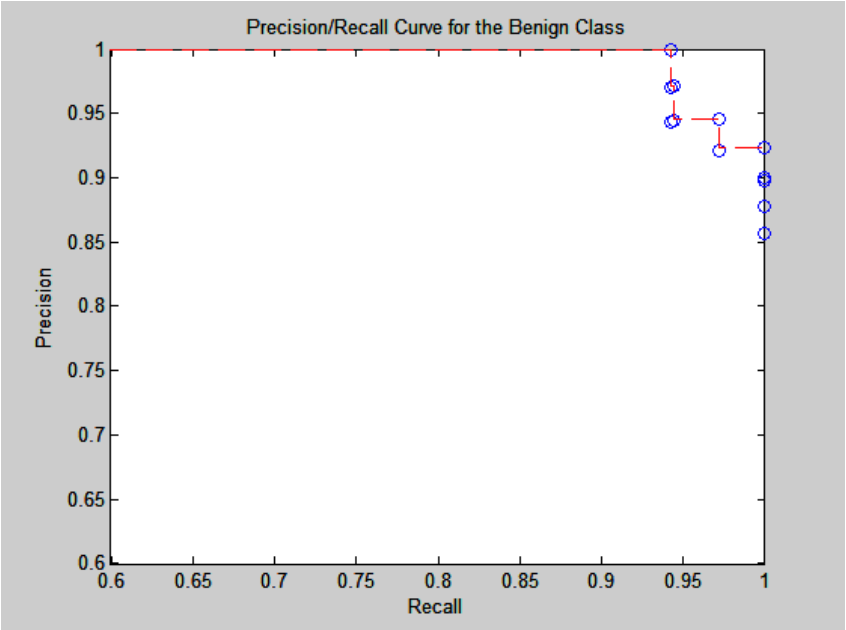


Figure 26. The Precision and Recall Curve for the k-Nearest Neighbor classifiers using the Density Sensitive Distance Measurement for the Benign Class in the Wisconsin Diagnostic Breast Cancer (WDBC) Dataset

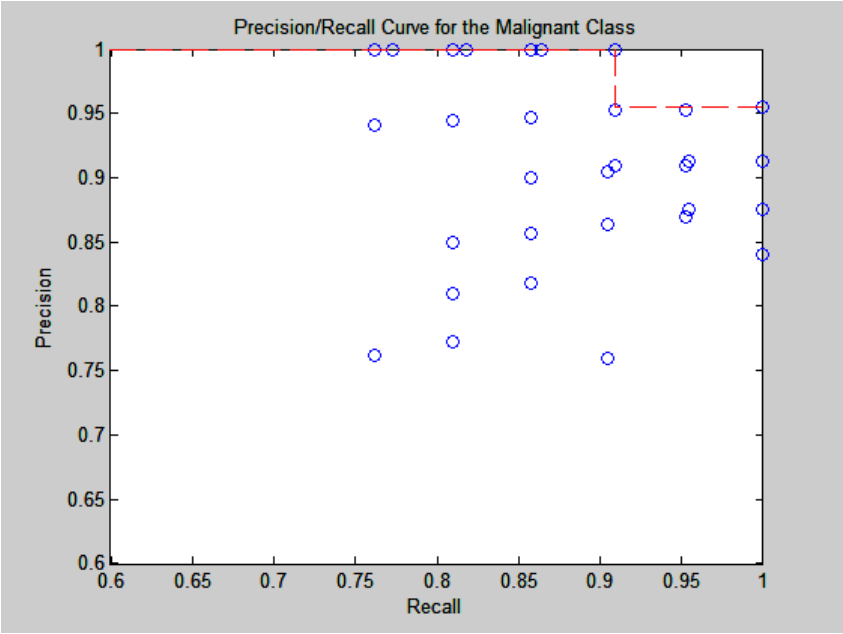


Figure 27. The Precision and Recall Curve for the k-Nearest Neighbor classifiers using Euclidean Distance for the Malignant Class in the Wisconsin Diagnostic Breast Cancer (WDBC) Dataset

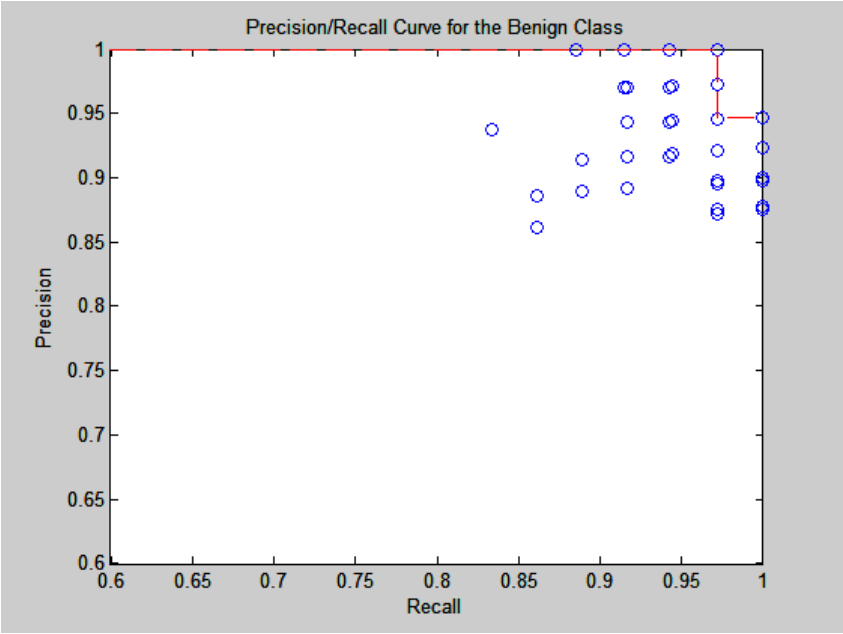


Figure 28. The Precision and Recall Curve for the k-Nearest Neighbor classifiers using Euclidean Distance for the Benign Class in the Wisconsin Diagnostic Breast Cancer (WDBC) Dataset

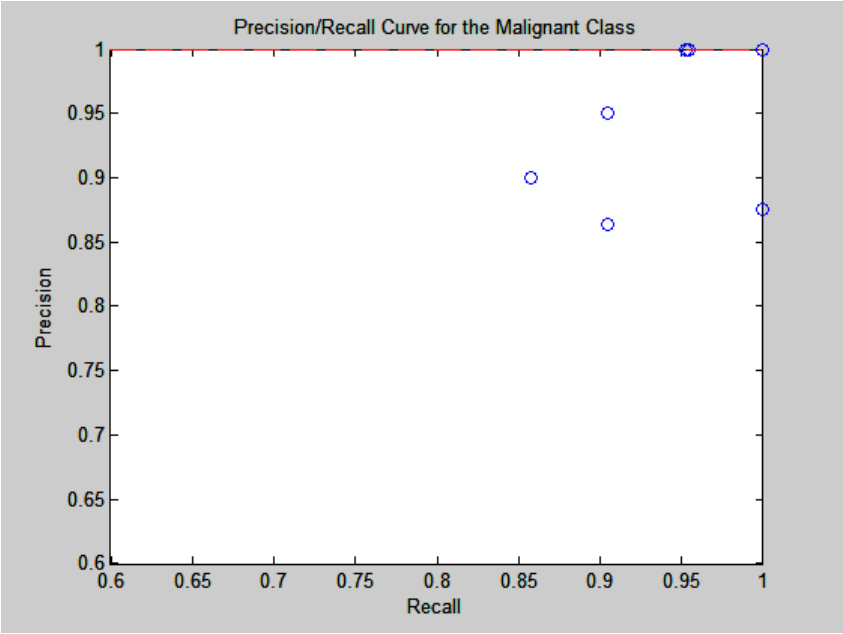


Figure 29. The Precision and Recall Curve for the Support Vector Machines for the Malignant Class in the Wisconsin Diagnostic Breast Cancer (WDBC) Dataset

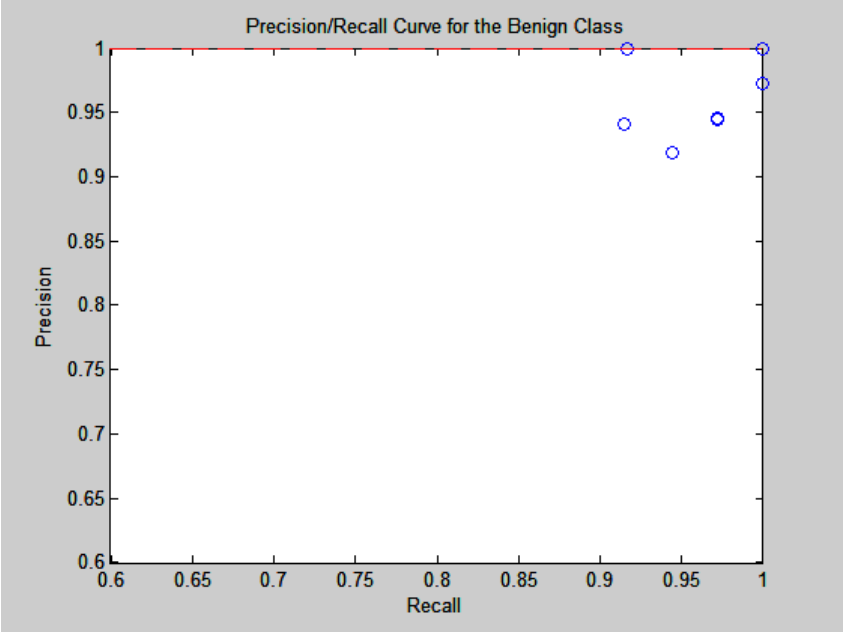


Figure 30. The Precision and Recall Curve for the Support Vector Machines for the Benign Class in the Wisconsin Diagnostic Breast Cancer (WDBC) Dataset

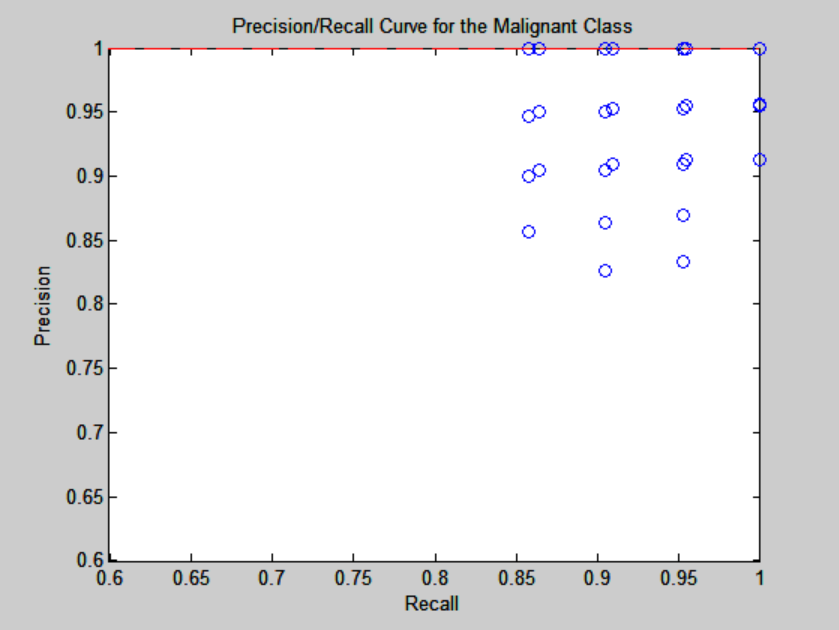


Figure 31. The Precision and Recall Curve for the Random Forests for the Malignant Class in the Wisconsin Diagnostic Breast Cancer (WDBC) Dataset

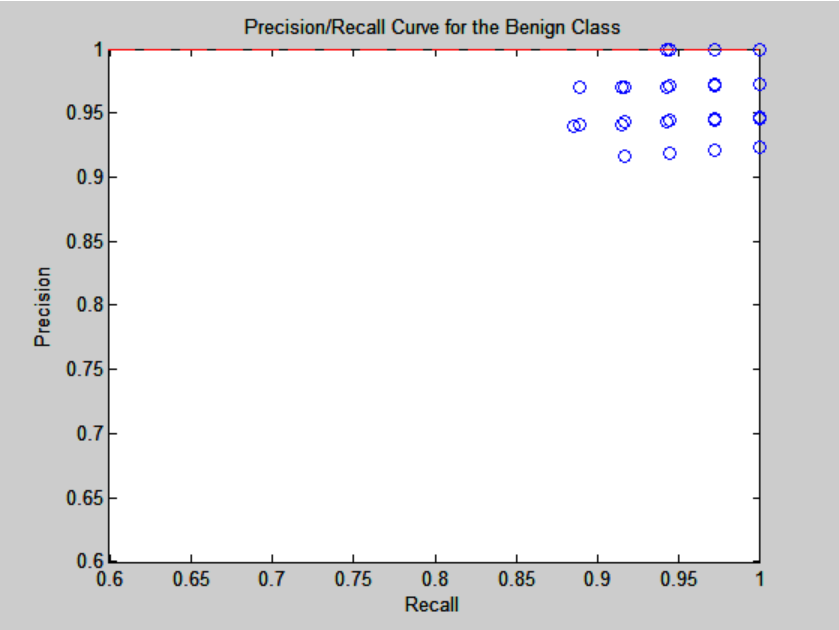


Figure 32. The Precision and Recall Curve for the Random Forests for the Benign Class in the Wisconsin Diagnostic Breast Cancer (WDBC) Dataset

4. Discussion

Since the proposed density sensitive distance measurement is essentially a locally weighted Euclidean distance, k-Nearest Neighbor classification using this density sensitive distance measurement slightly outperforms the same classifier using Euclidean distance. However, we note that the intervals (i.e., the means $\pm$ the standard deviations) overlap.

As expected, the modern supervised learning algorithm, Random Forest, dominates overall accuracy, overall error rate, and precision and recall for each class. Although the intervals for k-Nearest Neighbor using the density sensitive distance measurement and Random Forest do overlap, that overlap is quite slight.

B. THE ONES, TWOS, AND THREES FROM THE MNIST DATABASE OF HANDWRITTEN DIGITS

1. Overall Accuracy and Error Rate

Classifier	Overall Accuracy	Overall Error Rate
k-Nearest Neighbor using the Density Sensitive Distance Measurement with $k_{\text{classification}} = 1$ and $k_{\substack{\text{kernel density} \\ \text{estimation}}} = 150$	0.994105 ± 0.00168777	0.00589454 ± 0.00168777

k -Nearest Neighbor using the Density Sensitive Distance Measurement with $k_{\text{classification}} = 1$ and $k_{\substack{\text{kernel density} \\ \text{estimation}}} = 100$	0.994052 ± 0.00169415	0.00594762 ± 0.00169415
k -Nearest Neighbor using Euclidean Distance with $k = 1$	0.993575 ± 0.00179531	0.00642549 ± 0.00179531
k -Nearest Neighbor using Euclidean Distance with $k = 3$	0.992353 ± 0.00203715	0.00764703 ± 0.00203715
Support Vector Machine using RBF Kernel with $\gamma = 5.99484 \times 10^{-7}$ and $\nu = 0.1$	0.98837 ± 0.00364056	0.0116298 ± 0.00364056
Support Vector Machine using RBF Kernel with $\gamma = 10^{-7}$ and $\nu = 0.1$	0.981785 ± 0.00472434	0.0182148 ± 0.00472434
Random Forests with $\text{trees} = 20$, $\text{depth} = 20$, and $\text{split size} = 4$	0.985343 ± 0.00392884	0.0146569 ± 0.00392884

Random Forests with $\text{trees} = 20$, $\text{depth} = 20$, and $\text{split size} = 3$	0.985025 ± 0.00416513	0.0149753 ± 0.00416513

Table 5. Overall Accuracy and Error Rate for the Ones, Twos, and Threes from the MNIST Database of Handwritten Digits

2. Class Precision and Recall

	Class: Ones		**Class: Twos**		**Class: Threes**	
Classifier	**Precision**	**Recall**	**Precision**	**Recall**	**Precision**	**Recall**
k-Nearest Neighbor using the Density Sensitive Distance Measurement with $k_{\text{classification}} = 1$ and $k_{\substack{\text{kernel density} \\ \text{estimation}}} = 150$	0.995565 ± 0.00294162	0.99733 ± 0.00136235	0.991615 ± 0.00304218	0.991777 ± 0.00278905	0.994937 ± 0.00161508	0.992824 ± 0.00354039
k-Nearest Neighbor using the Density Sensitive Distance Measurement with $k_{\text{classification}} = 1$ and $k_{\substack{\text{kernel density} \\ \text{estimation}}} = 100$	0.995419 ± 0.00297786	0.99733 ± 0.00136235	0.991614 ± 0.0030415	0.991609 ± 0.00295768	0.994937 ± 0.00161508	0.992824 ± 0.00354039
k-Nearest Neighbor using Euclidean Distance with $k = 1$	0.993805 ± 0.00330286	0.997924 ± 0.00143185	0.992094 ± 0.00326101	0.989091 ± 0.00308645	0.994777 ± 0.00251895	0.99315 ± 0.00358793

k-Nearest Neighbor using Euclidean Distance with $k = 3$	0.989572 ± 0.0035872	0.99822 ± 0.00168445	0.992744 ± 0.00285545	0.986407 ± 0.00527022	0.995094 ± 0.00297708	0.991682 ± 0.00236131
Support Vector Machine using RBF Kernel with $\gamma = 5.99484 \times 10^{-7}$ and $\nu = 0.1$	0.997592 ± 0.001611	0.982498 ± 0.00702177	0.977642 ± 0.00713341	0.995805 ± 0.00276832	0.989069 ± 0.00539673	0.987604 ± 0.004156
Support Vector Machine using RBF Kernel with $\gamma = 10^{-7}$ and $\nu = 0.1$	0.987824 ± 0.00496109	0.985168 ± 0.00576701	0.974657 ± 0.00679761	0.986239 ± 0.00416655	0.982254 ± 0.00795159	0.97374 ± 0.00739521
Random Forests with $\text{trees} = 20$, $\text{depth} = 20$, and $\text{split size} = 4$	0.994931 ± 0.00322289	0.988876 ± 0.00485519	0.972341 ± 0.00813912	0.989427 ± 0.00371177	0.987798 ± 0.00305495	0.97749 ± 0.00752933
Random Forests with $\text{trees} = 20$, $\text{depth} = 20$, and $\text{split size} = 3$	0.994485 ± 0.0043141	0.988283 ± 0.00481462	0.972783 ± 0.00615367	0.988756 ± 0.00581317	0.986863 ± 0.00733193	0.977817 ± 0.00667079

Table 6. Precision and Recall for the Ones, Twos, and Threes from the MNIST Database of Handwritten Digits

3. Precision and Recall Curves

The following plots show precision versus recall for all 10-fold cross-validation runs of all classifiers. In these curves, the scale for the Precision and Recall axes range from 0.90 to 1.00, vice 0.00 to 1.00, to emphasize the results.

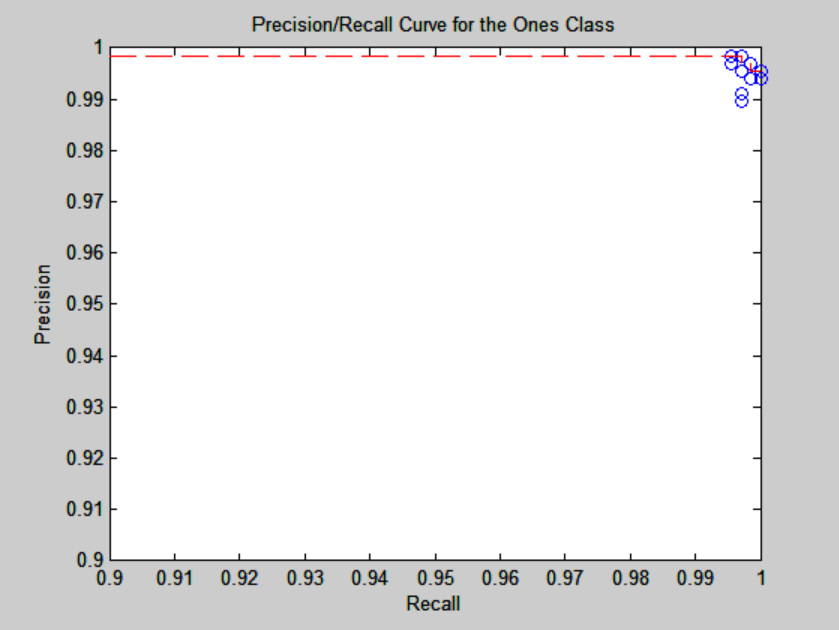


Figure 33. The Precision and Recall Curve for the k-Nearest Neighbor classifiers using the Density Sensitive Distance Measurement for the Ones Class of the MNIST Database of Handwritten Digits

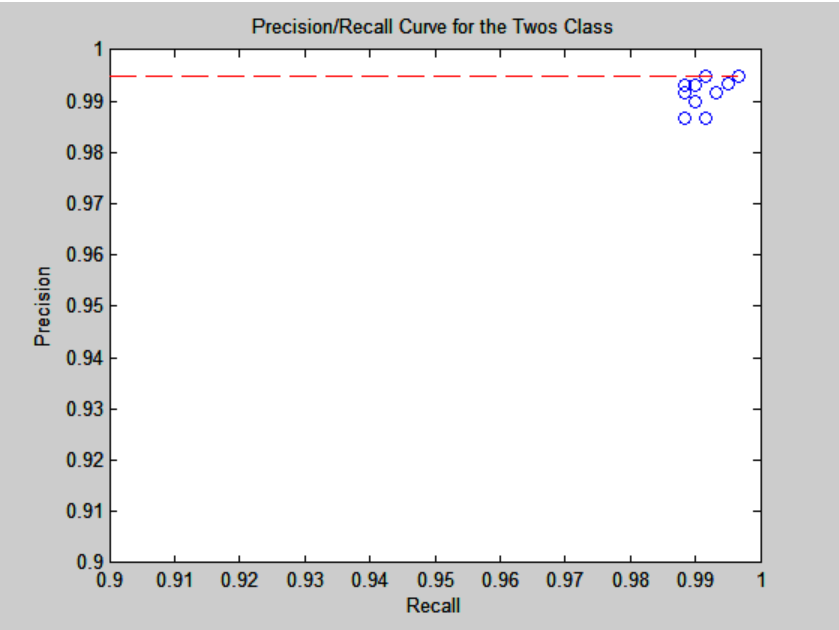


Figure 34. The Precision and Recall Curve for the k-Nearest Neighbor classifiers using the Density Sensitive Distance Measurement for the Twos Class of the MNIST Database of Handwritten Digits

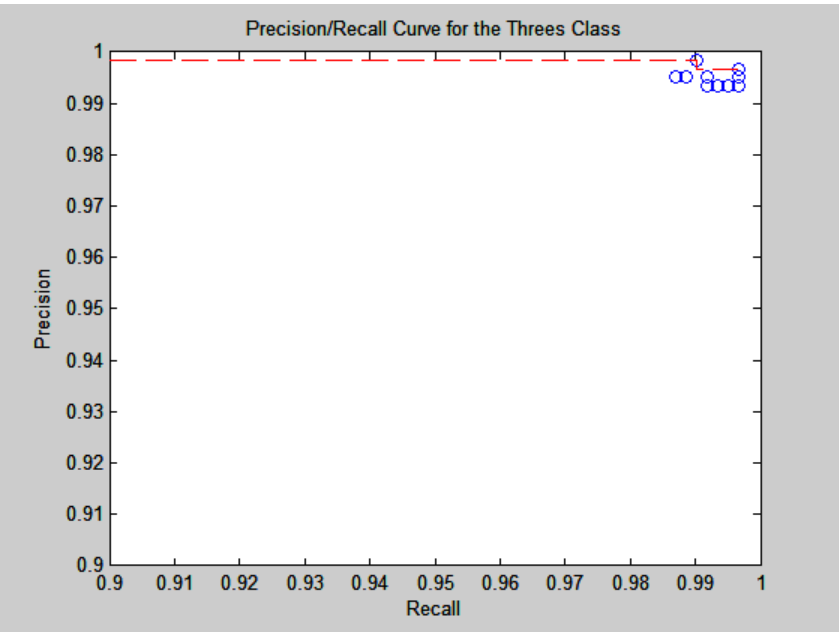


Figure 35. The Precision and Recall Curve for the k-Nearest Neighbor classifiers using the Density Sensitive Distance Measurement for the Threes Class of the MNIST Database of Handwritten Digits

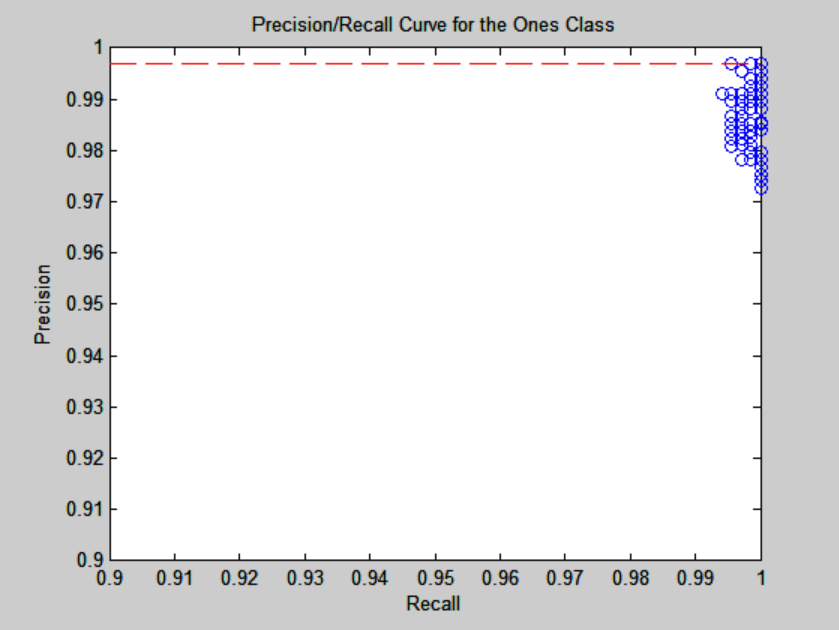


Figure 36. The Precision and Recall Curve for the k-Nearest Neighbor classifiers using Euclidean Distance for the Ones Class of the MNIST Database of Handwritten Digits

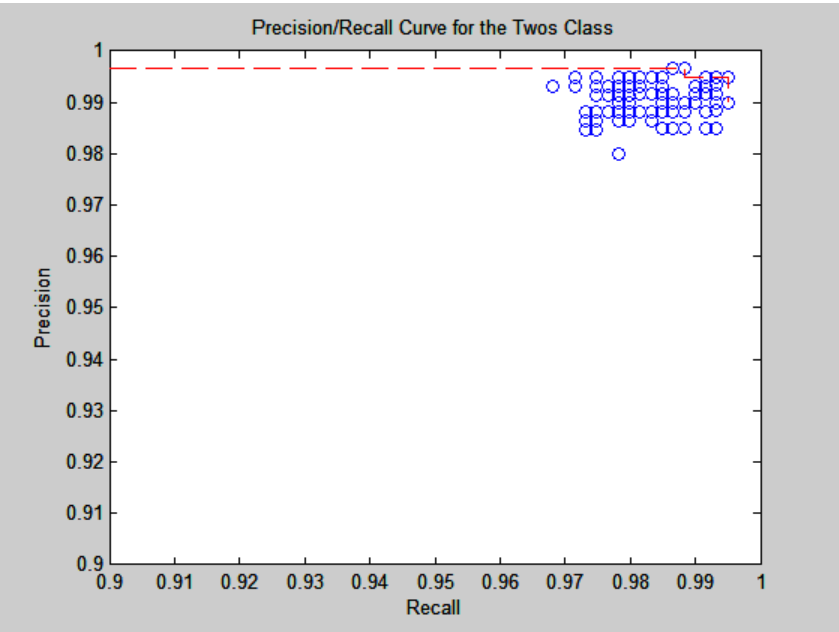


Figure 37. The Precision and Recall Curve for the k-Nearest Neighbor classifiers using Euclidean Distance for the Twos Class of the MNIST Database of Handwritten Digits

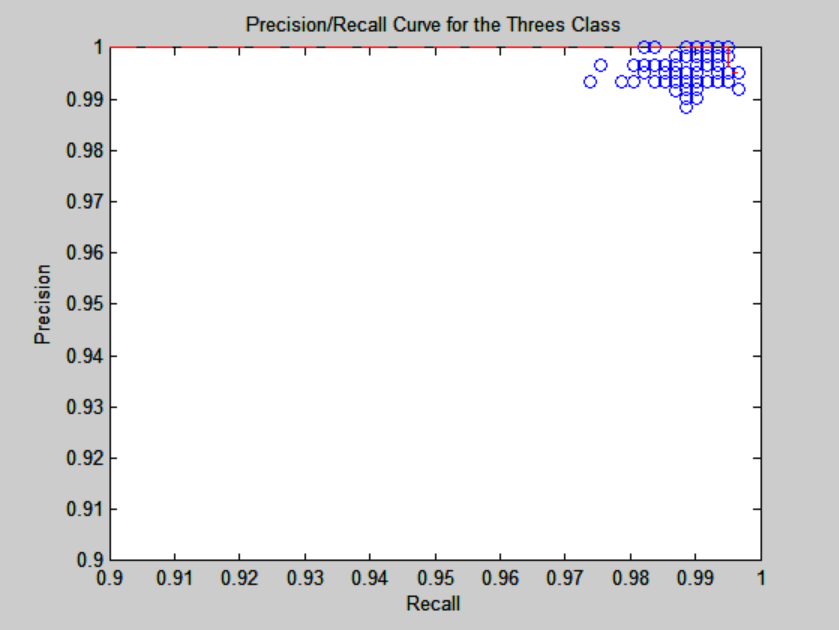


Figure 38. The Precision and Recall Curve for the k-Nearest Neighbor classifiers using Euclidean Distance for the Threes Class of the MNIST Database of Handwritten Digits

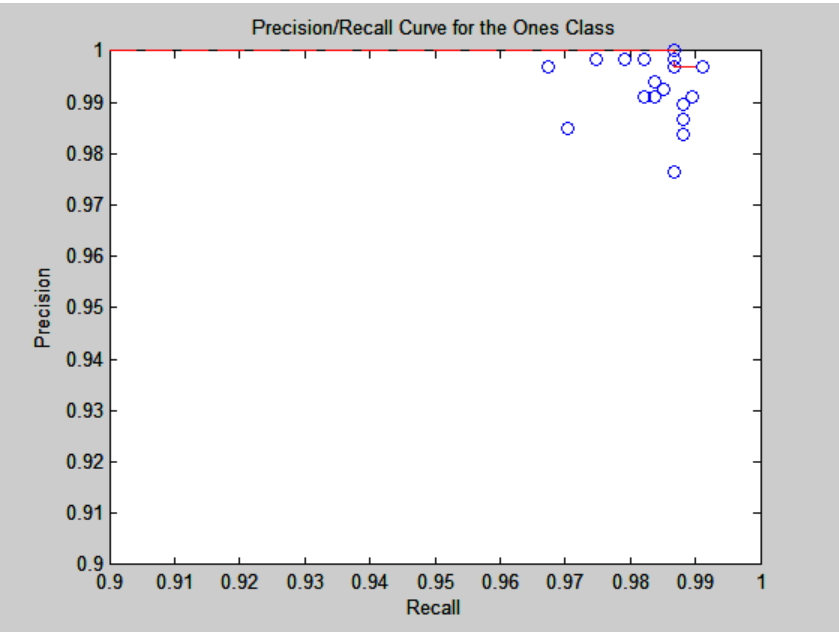


Figure 39. The Precision and Recall Curve for the Support Vector Machines for the Ones Class of the MNIST Database of Handwritten Digits

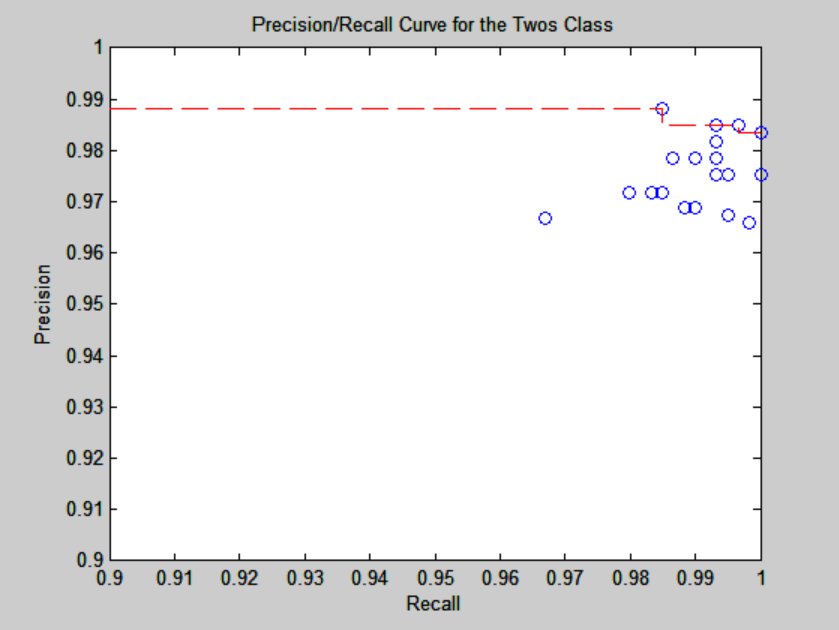


Figure 40. The Precision and Recall Curve for the Support Vector Machines for the Twos Class of the MNIST Database of Handwritten Digits

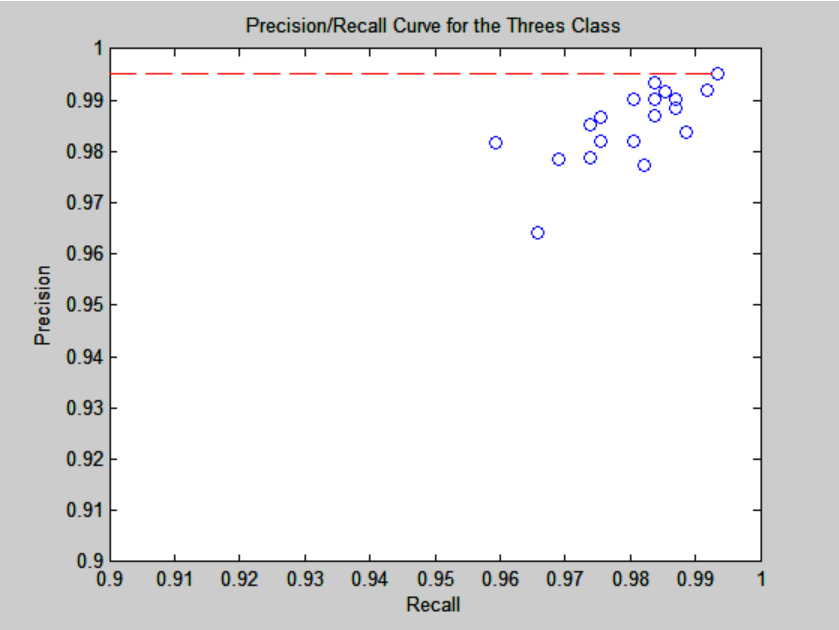


Figure 41. The Precision and Recall Curve for the Support Vector Machines for the Threes Class of the MNIST Database of Handwritten Digits

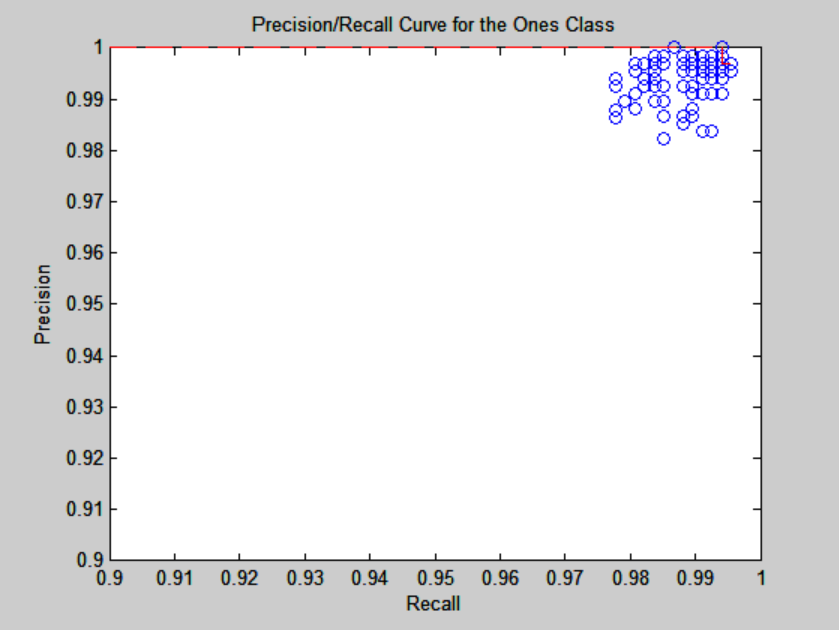


Figure 42. The Precision and Recall Curve for the Random Forests for the Ones Class of the MNIST Database of Handwritten Digits

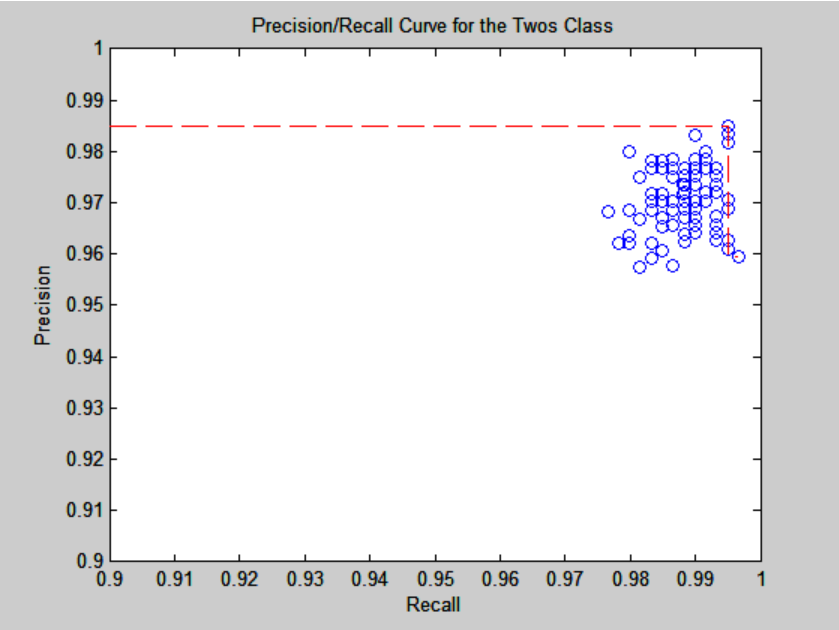


Figure 43. The Precision and Recall Curve for the Random Forests for the Twos Class of the MNIST Database of Handwritten Digits

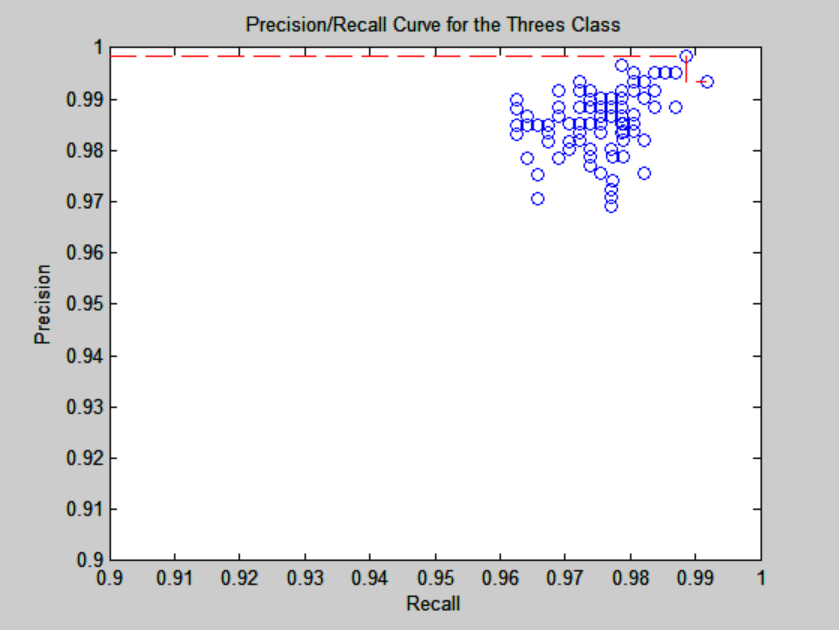


Figure 44. The Precision and Recall Curve for the Random Forests for the Threes Class of the MNIST Database of Handwritten Digits

4. Discussion

As with the WDBC dataset, k-Nearest Neighbor classification using the density sensitive distance measurement again slightly outperforms the same classifier using Euclidean distance. Similarly, we note that the intervals (i.e., the means $\pm$ the standard deviations) overlap.

However, the modern supervised learning classification algorithms, Support Vector Machine and Random Forests, do not dominate the overall accuracy and overall error rate. For both overall accuracy and error rate, the classifier using our density sensitive distance measurement has superior performance. Moreover, the intervals of the Support Vector Machines and Random Forests do not overlap with the classifier using our density sensitive distance measurement.

THIS PAGE INTENTIONALLY LEFT BLANK

VI. SUMMARY & CONCLUSIONS

A. SUMMARY

The proposed density sensitive distance measurement takes into account the density of each dataset over which it is used. This density sensitive distance measurement first finds the kernel density estimate of a given dataset and then takes the line integral along the surface of that kernel density estimate as we travel linearly from an initial position to a final position. The parameters required to be determined for this density sensitive distance measure are the kernel bandwidth and the scale. In this work, the kernel bandwidth is σ, the radius of the sphere each kernel approximates. Since we arbitrarily desire smooth kernel density estimates, we take advantage the additive properties of the chosen kernel when their centers are within 2σ of each other. Hence, we find the distance of the k-th nearest neighbor for each data point in a dataset and form a value for σ around these k-th nearest neighbors distances. The scale γ allows the variance in the kernel density estimate (i.e., the "vertical" variance) to be modified so that it will not be overpowered by the variances in the $\vec{\mathbf{x}}$ direction (i.e., "lateral" variances).

From the definition of the proposed density sensitive distance measurement, we utilized the density sensitive distance measurement in supervised learning in order to determine its utility and performance. Using stratified 10-fold cross validation to determine the generalization error, we trained and tested the k-Nearest Neighbor classifier using the proposed measurement. We also compared that classifier with k-Nearest Neighbor classification using Euclidean distance and two modern supervised learning algorithms, Support Vector Machines and Random Forests. This comparison took place over two datasets, the Wisconsin Diagnostic Breast Cancer (WDBC) dataset and a portion of the MNIST Database of Handwritten Digits.

B. CONCLUSIONS

The proposed density sensitive distance measure behaved as if it were a locally weighted Euclidean distance. As k-Nearest Neighbor classification using Euclidean distance did well, then k-Nearest Neighbor using our density sensitive distance did slightly better. During classification, when proximity was a nominal factor compared to density, as it was in the WDBC dataset, then our density sensitive distance measurement was nominally more successful than Euclidean distance, but subordinate to the modern algorithms involved in Support Vector Machine and Random Forests classification. When proximity was a larger factor in classification, as it was in the MNIST dataset, then our density sensitive distance measurement was superior to Support Vector Machines and Random Forests (although still only nominally better than Euclidean distance). All classifiers over both datasets did extremely well; therefore, future research using this density sensitive distance measurement for classification should concentrate on more difficult datasets.

The proposed density sensitive distance measurement conforms better to the shape of the data than Euclidean distance and performs slightly better in k-Nearest Neighbor classification; however, this density sensitive distance measurement comes at a high computation cost. Since the line integral must use values from the kernel density estimation, a single distance calculation must iterate over the entire training dataset hundreds (if not thousands of time) times. For small datasets, this can be negligible; however, for medium sized databases and beyond, this is may be too great a price to pay. To mitigate this computational cost, there are many approximations that can be made to substantially speed up the calculation of this density sensitive distance measurement.

C. FUTURE WORK

The proposed density sensitive distance measurement was used in classification on easier datasets; hence, all classifiers performed extremely well. Since all the classifiers had exceptional performance, it made a definitive comparison more challenging. In future comparisons, more discriminating datasets should be used.

Also, the current implementation of the proposed density sensitive distance measurement can be optimized to only take into account training points that are approximately near to any given testing point. Depending on the value of σ, many points in the training dataset may negligibly contribute to the value of the kernel density estimate at a given testing point. Work can be done to determine which training points contribute and which do not, perhaps similarly to how KD Trees determine the relevancy of points involved in a range or near neighbor query.

Moreover, since the proposed density sensitive distance measurement behaves as a locally weighted Euclidean distance and since Euclidean distance is orders of magnitude faster, a light-weight non-linear regression of the kernel density estimate that applies a weight to Euclidean distance may greatly increase the speed and utility of this density sensitive distance measurement while minimally impacting its accuracy.

THIS PAGE INTENTIONALLY LEFT BLANK

VII. APPENDIX

A. THEOREM: THE NORMALIZED FIRST APPROXIMATION TO THE TAYLOR SERIES EXPANSION OF THE UPPER-HALF OF THE $n+1$ DIMENSIONAL ELLIPSOID CENTERED AT $\left(\mu_1,\ldots,\mu_n,0\right)$ AND ROTATED IN A HYPERPLANE RESTRICTED TO THE FIRST n DIMENSIONS IS THE PROBABILITY DENSITY FUNCTION OF THE MULTIVARIATE NORMAL DISTRIBUTION

To prove this we will first prove this for the $n+1$ dimensional axis-aligned ellipsoid and then extend this to any $n+1$ dimensional ellipsoid that has been rotated on a n dimensional hyperplane.

1. The Upper-Half of the $n+1$ Dimensional Axis-Aligned Ellipsoid

Let $x_1,\ldots,x_n,y$ be variables aligned to the $n+1$ axes of $\mathbb{R}^{n+1}$. The equation for an axis-aligned ellipsoid centered at $\left(\mu_1,\ldots,\mu_n,0\right)$ where $\mu_j \in \mathbb{R}$ for $j=1,\ldots,n$ with radii $\left(\sigma_1,\ldots,\sigma_n,\sigma_{n+1}\right)$ along each of those $n+1$ axes where $\sigma_k \in \mathbb{R}$ such that $\sigma_k > 0$ for $k=1,\ldots,n+1$ is given by the following:

$$\frac{\left(x_1-\mu_1\right)^2}{\sigma_1^2}+\cdots+\frac{\left(x_n-\mu_n\right)^2}{\sigma_n^2}+\frac{\left(y-0\right)^2}{\sigma_{n+1}^2}=1$$

$$\left(\sum_{j=1}^{n}\frac{\left(x_j-\mu_j\right)^2}{\sigma_j^2}\right)+\frac{\left(y-0\right)^2}{\sigma_{n+1}^2}=1$$

Solving for y, we have the following:

$$\left(\sum_{j=1}^{n}\frac{\left(x_j-\mu_j\right)^2}{\sigma_j^2}\right)+\frac{\left(y-0\right)^2}{\sigma_{n+1}^2}=1$$

$$\frac{\left(y-0\right)^2}{\sigma_{n+1}^2}=1-\left(\sum_{j=1}^{n}\frac{\left(x_j-\mu_j\right)^2}{\sigma_j^2}\right)$$

$$\left(y-0\right)^2=\sigma_{n+1}^2\left(1-\left(\sum_{j=1}^{n}\frac{\left(x_j-\mu_j\right)^2}{\sigma_j^2}\right)\right)$$

$$y=\pm\sqrt{\sigma_{n+1}^2\left(1-\left(\sum_{j=1}^{n}\frac{\left(x_j-\mu_j\right)^2}{\sigma_j^2}\right)\right)}$$

$$y=\pm\sigma_{n+1}\sqrt{1-\left(\sum_{j=1}^{n}\frac{\left(x_j-\mu_j\right)^2}{\sigma_j^2}\right)}$$

If we restrict our focus to the upper-half of this axis-aligned ellipsoid (i.e., $y\geq 0$), then we have the following:

$$y_{\substack{\text{upper-half}\\ \text{axis-aligned}\\ \text{ellipsoid}}}=+\sigma_{n+1}\sqrt{1-\left(\sum_{j=1}^{n}\frac{\left(x_j-\mu_j\right)^2}{\sigma_j^2}\right)}$$

Since $\underbrace{\lim_{u\to 0^+}\log\left(u\right)=-\infty}_{\text{from real analysis}}$ and $\underbrace{\lim_{u\to 0^-}\log\left(u\right)=-\infty}_{\text{from complex analysis}}$ for $u\in\mathbb{R}$, then $\lim_{u\to 0}\log\left(u\right)=-\infty$. Furthermore, since $\lim_{u\to 0}\log\left(u\right)=-\infty$ and $\lim_{v\to-\infty}\exp\left(v\right)=0$ for $u,v\in\mathbb{R}$, then $u=\exp\left(\log\left(u\right)\right)$ for $u\geq 0$. Since $u=\exp\left(\log\left(u\right)\right)$ for $u\in\mathbb{R}$ and $u\geq 0$, then we have the following:

$$y_{\substack{\text{upper-half}\\ \text{axis-aligned}\\ \text{ellipsoid}}} = \sigma_{n+1}\sqrt{1-\left(\sum_{j=1}^{n}\frac{\left(x_j-\mu_j\right)^2}{\sigma_j^2}\right)}$$

$$\exp\left(\log\left(y_{\substack{\text{upper-half}\\ \text{axis-aligned}\\ \text{ellipsoid}}}\right)\right) = \exp\left(\log\left(\sigma_{n+1}\sqrt{1-\left(\sum_{j=1}^{n}\frac{\left(x_j-\mu_j\right)^2}{\sigma_j^2}\right)}\right)\right)$$

$$y_{\substack{\text{upper-half}\\ \text{axis-aligned}\\ \text{ellipsoid}}} = \exp\left(\log\left(\sigma_{n+1}\right)+\log\left(\sqrt{1-\left(\sum_{j=1}^{n}\frac{\left(x_j-\mu_j\right)^2}{\sigma_j^2}\right)}\right)\right)$$

$$y_{\substack{\text{upper-half}\\ \text{axis-aligned}\\ \text{ellipsoid}}} = \exp\left(\log\left(\sigma_{n+1}\right)\right)\exp\left(\log\left(\left(1-\left(\sum_{j=1}^{n}\frac{\left(x_j-\mu_j\right)^2}{\sigma_j^2}\right)\right)^{\frac{1}{2}}\right)\right)$$

$$y_{\substack{\text{upper-half}\\ \text{axis-aligned}\\ \text{ellipsoid}}} = \sigma_{n+1}\exp\left(\frac{1}{2}\log\left(1-\left(\sum_{j=1}^{n}\frac{\left(x_j-\mu_j\right)^2}{\sigma_j^2}\right)\right)\right)$$

Since the Taylor series expansion of $f\left(u\right)$ at u_0 is

$f\left(u\right)=\sum_{i=0}^{\infty}\frac{f^{(i)}\left(u_0\right)}{i!}\left(u-u_0\right)^i$, then for $f\left(u\right)=\log\left(u\right)$ at $u_0=1$, we have

$f^{(0)}\left(u\right)=\log\left(u\right),\ f^{(m)}\left(u\right)=\frac{d^m}{du^m}\left[\log\left(u\right)\right]=\frac{\left(-1\right)^{\left(m-1\right)}\left(m-1\right)!}{\left(u\right)^m}$ for $m=1,2,3,\ldots$, and

the following:

$$
\begin{aligned}
\log\left(u\right) &= \sum_{i=0}^{\infty} \frac{f^{(i)}\left(u_0\right)}{i\,!}\left(u-u_0\right)^i \\
&= \frac{f^{(0)}\left(u_0\right)}{0\,!}\left(u-u_0\right)^0 + \sum_{i=1}^{\infty} \frac{f^{(i)}\left(u_0\right)}{i\,!}\left(u-u_0\right)^i \\
&= \log\left(1\right) + \sum_{i=1}^{\infty} \frac{\left(\dfrac{\left(-1\right)^{(i-1)}\left(i-1\right)!}{\left(1\right)^i}\right)}{i\,!}\left(u-1\right)^i \\
&= 0 + \sum_{i=1}^{\infty} \frac{\left(\left(-1\right)^{(i-1)}\left(i-1\right)!\right)}{i\,!}\left(u-1\right)^i \\
&= \sum_{i=1}^{\infty}\left(-1\right)^{(i-1)} \frac{\left(u-1\right)^i}{i}
\end{aligned}
$$

Substituting $u = 1 - \left(\sum_{j=1}^{n} \frac{\left(x_j - \mu_j\right)^2}{\sigma_j^2}\right)$, we have the following:

$$\log\left(1-\left(\sum_{j=1}^{n}\frac{\left(x_j-\mu_j\right)^2}{\sigma_j^2}\right)\right)=\sum_{i=1}^{\infty}\left(-1\right)^{\left(i-1\right)}\frac{\left(\left(1-\left(\sum_{j=1}^{n}\frac{\left(x_j-\mu_j\right)^2}{\sigma_j^2}\right)\right)-1\right)^i}{i}$$

$$=\sum_{i=1}^{\infty}\left(-1\right)^{\left(i-1\right)}\frac{\left(-\left(\sum_{j=1}^{n}\frac{\left(x_j-\mu_j\right)^2}{\sigma_j^2}\right)\right)^i}{i}$$

$$=\sum_{i=1}^{\infty}\left(-1\right)^{\left(i-1\right)}\frac{\left(-1\right)^i\left(\sum_{j=1}^{n}\frac{\left(x_j-\mu_j\right)^2}{\sigma_j^2}\right)^i}{i}$$

$$=\sum_{i=1}^{\infty}\left(-1\right)^{\left(2i-1\right)}\frac{\left(\sum_{j=1}^{n}\frac{\left(x_j-\mu_j\right)^2}{\sigma_j^2}\right)^i}{i}$$

$$=-\sum_{i=1}^{\infty}\frac{\left(\sum_{j=1}^{n}\frac{\left(x_j-\mu_j\right)^2}{\sigma_j^2}\right)^i}{i}$$

Returning to the upper-half our axis-aligned ellipsoid, we have the following:

$$y_{\substack{\text{upper-half}\\\text{axis-aligned}\\\text{ellipsoid}}}=\sigma_{n+1}\exp\left(\frac{1}{2}\log\left(1-\left(\sum_{j=1}^{n}\frac{\left(x_j-\mu_j\right)^2}{\sigma_j^2}\right)\right)\right)$$

$$=\sigma_{n+1}\exp\left(\frac{1}{2}\left(-\sum_{i=1}^{\infty}\frac{\left(\sum_{j=1}^{n}\frac{\left(x_j-\mu_j\right)^2}{\sigma_j^2}\right)^i}{i}\right)\right)$$

$$=\sigma_{n+1}\exp\left(-\sum_{i=1}^{\infty}\frac{\left(\sum_{j=1}^{n}\frac{\left(x_j-\mu_j\right)^2}{\sigma_j^2}\right)^i}{2i}\right)$$

Hence, the first approximation to the Taylor series expansion of the upper-half of our axis-aligned ellipsoid is the following:

$$
\begin{aligned}
y_{\substack{\text{upper-half}\\ \text{axis-aligned}\\ \text{ellipsoid}}} &= \sigma_{n+1} \exp\left(-\sum_{i=1}^{\infty} \frac{\left(\sum_{j=1}^{n} \frac{\left(x_j - \mu_j \right)^2}{\sigma_j^2} \right)^i}{2i} \right) \\
y_{\substack{\text{first approximation}\\ \text{to upper-half axis-}\\ \text{aligned ellipsoid}}} &= \sigma_{n+1} \exp\left(-\sum_{i=1}^{1} \frac{\left(\sum_{j=1}^{n} \frac{\left(x_j - \mu_j \right)^2}{\sigma_j^2} \right)^i}{2i} \right) \\
&= \sigma_{n+1} \exp\left(-\left(\frac{\left(\sum_{j=1}^{n} \frac{\left(x_j - \mu_j \right)^2}{\sigma_j^2} \right)^{(1)}}{2(1)} \right) \right) \\
&= \sigma_{n+1} \exp\left(-\frac{1}{2} \sum_{j=1}^{n} \frac{\left(x_j - \mu_j \right)^2}{\sigma_j^2} \right)
\end{aligned}
$$

We now parameterize σ_{n+1} using σ_j for $j = 1,\ldots,n$ such that the first approximation to the Taylor series expansion of the upper-half of our axis-aligned ellipsoid is normalized. In order for $y_{\substack{\text{first approximation}\\ \text{to upper-half axis-}\\ \text{aligned ellipsoid}}}$ to be normalized, the "area" under its "curve" needs to be one; in other words, we need the following:

$$
\begin{aligned}
&\int_{-\infty}^{\infty} \cdots \int_{-\infty}^{\infty} y_{\substack{\text{first approximation}\\ \text{to upper-half axis-}\\ \text{aligned ellipsoid}}} dx_1 \cdots dx_n = 1 \\
&\int_{-\infty}^{\infty} \cdots \int_{-\infty}^{\infty} \left(\sigma_{n+1} \exp\left(-\frac{1}{2} \sum_{j=1}^{n} \frac{\left(x_j - \mu_j \right)^2}{\sigma_j^2} \right) \right) dx_1 \cdots dx_n = 1
\end{aligned}
$$

For this to occur, we have the following:

$$\int_{-\infty}^{\infty}\cdots\int_{-\infty}^{\infty}\sigma_{n+1}\exp\left(-\frac{1}{2}\sum_{j=1}^{n}\frac{\left(x_j-\mu_j\right)^2}{\sigma_j^2}\right)dx_1\cdots dx_n=1$$

$$\int_{-\infty}^{\infty}\cdots\int_{-\infty}^{\infty}\sigma_{n+1}\exp\left(-\frac{\left(x_1-\mu_1\right)^2}{2\sigma_1^2}\cdots-\frac{\left(x_n-\mu_n\right)^2}{2\sigma_n^2}\right)dx_1\cdots dx_n=1$$

$$\int_{-\infty}^{\infty}\cdots\int_{-\infty}^{\infty}\sigma_{n+1}\exp\left(-\frac{\left(x_1-\mu_1\right)^2}{2\sigma_1^2}\right)\cdots\exp\left(-\frac{\left(x_n-\mu_n\right)^2}{2\sigma_n^2}\right)dx_1\cdots dx_n=1$$

$$\sigma_{n+1}\int_{-\infty}^{\infty}\exp\left(-\frac{\left(x_1-\mu_1\right)^2}{2\sigma_1^2}\right)dx_1\cdots\int_{-\infty}^{\infty}\exp\left(-\frac{\left(x_n-\mu_n\right)^2}{2\sigma_n^2}\right)dx_n=1$$

$$\sigma_{n+1}\left(\sqrt{2\pi}\sigma_1\right)\cdots\left(\sqrt{2\pi}\sigma_n\right)=1$$

$$\sigma_{n+1}\left(\sqrt{2\pi}\right)^n\prod_{k=1}^{n}\sigma_k=1$$

$$\sigma_{n+1}=\frac{1}{\left(\sqrt{2\pi}\right)^n\prod_{k=1}^{n}\sigma_k}$$

$$\sigma_{n+1}=\frac{1}{\left(2\pi\right)^{\frac{n}{2}}\prod_{k=1}^{n}\sigma_k}$$

Therefore, the normalized first approximation to the Taylor series expansion of the upper-half of our axis-aligned ellipsoid is the following:

$$y_{\substack{\text{normalized}\\ \text{first approximation}\\ \text{to upper-half axis-}\\ \text{aligned ellipsoid}}}=\frac{1}{\left(2\pi\right)^{\frac{n}{2}}\prod_{k=1}^{n}\sigma_k}\exp\left(-\frac{1}{2}\sum_{j=1}^{n}\frac{\left(x_j-\mu_j\right)^2}{\sigma_j^2}\right)$$

Note: That the normalized first approximation to the Taylor series expansion of the upper-half of our axis-aligned ellipsoid is the probability density function of the axis-aligned multivariate Normal distribution.

Let $\vec{\vec{\mathbf{V}}}$ (for variances) be the diagonal matrix created from the square of the first n radii such that we have the following:

$$\vec{\vec{\mathbf{V}}} = \begin{bmatrix} \sigma_1^2 & 0 & 0 \\ 0 & \ddots & 0 \\ 0 & 0 & \sigma_n^2 \end{bmatrix},$$

then $\vec{\vec{\mathbf{V}}}$ has the following properties:

1) $\left|\vec{\vec{\mathbf{V}}}\right| = \prod_{k=1}^{n} \sigma_k^2$

2) $\left|\vec{\vec{\mathbf{V}}}\right|^{\frac{1}{2}} = \left(\prod_{k=1}^{n} \sigma_k^2\right)^{\frac{1}{2}} = \prod_{k=1}^{n} \sigma_k$

3) $\vec{\vec{\mathbf{V}}}^{-1} = \begin{bmatrix} \frac{1}{\sigma_1^2} & 0 & 0 \\ 0 & \ddots & 0 \\ 0 & 0 & \frac{1}{\sigma_n^2} \end{bmatrix}$

where $\left|\vec{\vec{\mathbf{V}}}\right| > 0$ since $\sigma_k > 0$ for $k = 1,\ldots,n$.

With these properties in mind, we can re-write the normalized first approximation to the Taylor series expansion of the upper-half of our axis-aligned ellipsoid as the following:

$$\begin{aligned} y_{\substack{\text{normalized} \\ \text{first approximation} \\ \text{to upper-half axis-} \\ \text{aligned ellipsoid}}} &= \frac{1}{\left(2\pi\right)^{\frac{n}{2}} \prod_{k=1}^{n} \sigma_k} \exp\left(-\frac{1}{2} \sum_{j=1}^{n} \frac{\left(x_j - \mu_j\right)^2}{\sigma_j^2}\right) \\ &= \frac{1}{\left(2\pi\right)^{\frac{n}{2}} \left|\vec{\vec{\mathbf{V}}}\right|^{\frac{1}{2}}} \exp\left(-\frac{1}{2}\left(\vec{\mathbf{x}} - \vec{\boldsymbol{\mu}}\right)^T \vec{\vec{\mathbf{V}}}^{-1} \left(\vec{\mathbf{x}} - \vec{\boldsymbol{\mu}}\right)\right) \end{aligned}$$

where $\vec{\mathbf{x}} = \left(x_1,\ldots x_n\right)^T$ and $\vec{\boldsymbol{\mu}} = \left(\mu_1,\ldots,\mu_n\right)^T$; in other words, $\vec{\mathbf{x}}$ and $\vec{\boldsymbol{\mu}}$ are column vectors. Moreover, since all the radii are axis aligned, then there is no covariance and

$\ddot{\vec{\mathbf{V}}} = \ddot{\vec{\mathbf{\Sigma}}}_{\text{axis-aligned}}$ where $\ddot{\vec{\mathbf{\Sigma}}}_{\text{axis-aligned}}$ is the axis aligned covariance matrix (i.e., positive non-zero values only on the diagonal); hence, we can re-write the normalized first approximation to the Taylor series expansion of the upper-half of our axis-aligned ellipsoid as the following:

$$y_{\substack{\text{normalized first approximation to upper-half axis-aligned ellipsoid}}} = \frac{1}{\left(2\pi\right)^{\frac{n}{2}} \left|\ddot{\vec{\mathbf{\Sigma}}}_{\text{axis-aligned}}\right|^{\frac{1}{2}}} \exp\left(-\frac{1}{2}\left(\vec{\mathbf{x}}-\vec{\boldsymbol{\mu}}\right)^{T} \ddot{\vec{\mathbf{\Sigma}}}_{\text{axis-aligned}}^{-1}\left(\vec{\mathbf{x}}-\vec{\boldsymbol{\mu}}\right)\right)$$

2. The Upper-Half of the $n+1$ Dimensional Rotated Ellipsoid

Since the equation for our $n+1$ dimensional axis-aligned ellipsoid is the following:

$$\frac{\left(x_1-\mu_1\right)^2}{\sigma_1^2}+\cdots+\frac{\left(x_n-\mu_n\right)^2}{\sigma_n^2}+\frac{\left(y-0\right)^2}{\sigma_{n+1}^2}=1$$

then the upper-half axis-aligned ellipsoid can be re-written as the following:

$$\begin{aligned}\left(\vec{\mathbf{x}}-\vec{\boldsymbol{\mu}}\right)^{T} \ddot{\vec{\mathbf{V}}}^{-1}\left(\vec{\mathbf{x}}-\vec{\boldsymbol{\mu}}\right)+\frac{y^2}{\sigma_{n+1}^2} &= 1 \\ y &= \pm\sigma_{n+1}\sqrt{1-\left(\vec{\mathbf{x}}-\vec{\boldsymbol{\mu}}\right)^{T} \ddot{\vec{\mathbf{V}}}^{-1}\left(\vec{\mathbf{x}}-\vec{\boldsymbol{\mu}}\right)} \\ y_{\substack{\text{upper-half axis-aligned ellipsoid}}} &= +\sigma_{n+1}\sqrt{1-\left(\vec{\mathbf{x}}-\vec{\boldsymbol{\mu}}\right)^{T} \ddot{\vec{\mathbf{V}}}^{-1}\left(\vec{\mathbf{x}}-\vec{\boldsymbol{\mu}}\right)}\end{aligned}$$

Let $\ddot{\vec{\mathbf{R}}}_{i,j}$ be the rotation matrix that rotates θ radians in the hyperplane spanned by $\vec{\mathbf{x}}_i$ and $\vec{\mathbf{x}}_j$ basis vectors where $\vec{\mathbf{x}}_i, \vec{\mathbf{x}}_j \in \mathbb{R}^n$, $i = 1,\ldots,n$, $j = 1,\ldots,n$, and $i \neq j$; hence, we have the following:

$$\vec{\vec{\mathbf{R}}}_{i,j}=\begin{bmatrix}
1 & 0 & \cdots & & & & & & & \cdots & 0 & 0\\
0 & 1 & 0 & \cdots & & & & & & \cdots & & 0\\
\vdots & 0 & \ddots & \ddots & & & & & & & \vdots & \vdots\\
 & \vdots & \ddots & 1 & & & & & & & & \\
 & & & & \cos\left(\theta\right) & & & \cos\left(\theta+\frac{\pi}{2}\right) & & & & \\
 & & & & & 1 & & & & & & \\
 & & & & & & \ddots & & & & & \\
 & & & & & & & 1 & & & & \\
 & & & & \sin\left(\theta\right) & & & \sin\left(\theta+\frac{\pi}{2}\right) & & & & \\
 & & & & & & & & 1 & \ddots & \vdots & \\
\vdots & \vdots & & & & & & & \ddots & \ddots & 0 & \vdots\\
0 & & \cdots & & & & & & \cdots & 0 & 1 & 0\\
0 & 0 & \cdots & & & & & & & \cdots & 0 & 1
\end{bmatrix}$$

As a rotation matrix, $\vec{\vec{\mathbf{R}}}_{i,j}$ has the following properties:

1) $\vec{\vec{\mathbf{R}}}_{i,j}$ is an orthogonal matrix

2) $\vec{\vec{\mathbf{R}}}_{i,j}{}^{T}=\vec{\vec{\mathbf{R}}}_{i,j}{}^{-1}$

3) $\left|\vec{\vec{\mathbf{R}}}_{i,j}\right|=\pm1$

Let $\vec{\vec{\mathbf{R}}}$ represent all possible rotations in $\mathbb{R}^n$, then we have the following:

$$\vec{\vec{\mathbf{R}}}=\underbrace{\vec{\vec{\mathbf{R}}}_{1,2}\cdots\vec{\vec{\mathbf{R}}}_{1,n}\vec{\vec{\mathbf{R}}}_{2,3}\cdots\vec{\vec{\mathbf{R}}}_{2,n}\vec{\vec{\mathbf{R}}}_{3,4}\cdots\vec{\vec{\mathbf{R}}}_{3,n}\cdots\vec{\vec{\mathbf{R}}}_{n-1,n}}_{\sum_{i=1}^{n-1}\sum_{j=i+1}^{n}1=\frac{(n-1)((n-1)+1)}{2}=\frac{n(n-1)}{2}\text{ rotation matrices}}$$

Moreover, $\vec{\vec{\mathbf{R}}}$ has the following properties:

1) Since the product of orthogonal matrices is an orthogonal matrix, then $\vec{\vec{\mathbf{R}}}$ is an orthogonal matrix

2) Since $\vec{\vec{\mathbf{R}}} = \vec{\vec{\mathbf{R}}}_{1,2} \cdots \vec{\vec{\mathbf{R}}}_{1,n} \vec{\vec{\mathbf{R}}}_{2,3} \cdots \vec{\vec{\mathbf{R}}}_{2,n} \vec{\vec{\mathbf{R}}}_{3,4} \cdots \vec{\vec{\mathbf{R}}}_{3,n} \cdots \vec{\vec{\mathbf{R}}}_{n-1,n}$, then we have the following:

$$\begin{aligned}
\vec{\vec{\mathbf{R}}}^T &= \left(\vec{\vec{\mathbf{R}}}_{1,2} \cdots \vec{\vec{\mathbf{R}}}_{1,n} \vec{\vec{\mathbf{R}}}_{2,3} \cdots \vec{\vec{\mathbf{R}}}_{2,n} \vec{\vec{\mathbf{R}}}_{3,4} \cdots \vec{\vec{\mathbf{R}}}_{3,n} \cdots \vec{\vec{\mathbf{R}}}_{n-1,n} \right)^T \\
&= \vec{\vec{\mathbf{R}}}^T_{n-1,n} \cdots \vec{\vec{\mathbf{R}}}^T_{3,n} \cdots \vec{\vec{\mathbf{R}}}^T_{3,4} \vec{\vec{\mathbf{R}}}^T_{2,n} \cdots \vec{\vec{\mathbf{R}}}^T_{2,3} \vec{\vec{\mathbf{R}}}^T_{1,n} \cdots \vec{\vec{\mathbf{R}}}^T_{1,2} \\
&= \vec{\vec{\mathbf{R}}}^{-1}_{n-1,n} \cdots \vec{\vec{\mathbf{R}}}^{-1}_{3,n} \cdots \vec{\vec{\mathbf{R}}}^{-1}_{3,4} \vec{\vec{\mathbf{R}}}^{-1}_{2,n} \cdots \vec{\vec{\mathbf{R}}}^{-1}_{2,3} \vec{\vec{\mathbf{R}}}^{-1}_{1,n} \cdots \vec{\vec{\mathbf{R}}}^{-1}_{1,2} \\
&= \left(\vec{\vec{\mathbf{R}}}_{1,2} \cdots \vec{\vec{\mathbf{R}}}_{1,n} \vec{\vec{\mathbf{R}}}_{2,3} \cdots \vec{\vec{\mathbf{R}}}_{2,n} \vec{\vec{\mathbf{R}}}_{3,4} \cdots \vec{\vec{\mathbf{R}}}_{3,n} \cdots \vec{\vec{\mathbf{R}}}_{n-1,n} \right)^{-1} \\
&= \vec{\vec{\mathbf{R}}}^{-1}
\end{aligned}$$

3) Since $\vec{\vec{\mathbf{R}}} = \vec{\vec{\mathbf{R}}}_{1,2} \cdots \vec{\vec{\mathbf{R}}}_{1,n} \vec{\vec{\mathbf{R}}}_{2,3} \cdots \vec{\vec{\mathbf{R}}}_{2,n} \vec{\vec{\mathbf{R}}}_{3,4} \cdots \vec{\vec{\mathbf{R}}}_{3,n} \cdots \vec{\vec{\mathbf{R}}}_{n-1,n}$, then we have the following:

$$\begin{aligned}
\left| \vec{\vec{\mathbf{R}}} \right| &= \left| \vec{\vec{\mathbf{R}}}_{1,2} \cdots \vec{\vec{\mathbf{R}}}_{1,n} \vec{\vec{\mathbf{R}}}_{2,3} \cdots \vec{\vec{\mathbf{R}}}_{2,n} \vec{\vec{\mathbf{R}}}_{3,4} \cdots \vec{\vec{\mathbf{R}}}_{3,n} \cdots \vec{\vec{\mathbf{R}}}_{n-1,n} \right| \\
&= \left| \vec{\vec{\mathbf{R}}}_{1,2} \right| \cdots \left| \vec{\vec{\mathbf{R}}}_{1,n} \right| \left| \vec{\vec{\mathbf{R}}}_{2,3} \right| \cdots \left| \vec{\vec{\mathbf{R}}}_{2,n} \right| \left| \vec{\vec{\mathbf{R}}}_{3,4} \right| \cdots \left| \vec{\vec{\mathbf{R}}}_{3,n} \right| \cdots \left| \vec{\vec{\mathbf{R}}}_{n-1,n} \right| \\
&= \pm 1
\end{aligned}$$

With these properties in mind, we can rotate our upper-half ellipsoid around its center by multiplying $\vec{\vec{\mathbf{R}}}^T_{n-1,n} \cdots \vec{\vec{\mathbf{R}}}^T_{3,n} \cdots \vec{\vec{\mathbf{R}}}^T_{3,4} \vec{\vec{\mathbf{R}}}^T_{2,n} \cdots \vec{\vec{\mathbf{R}}}^T_{2,3} \vec{\vec{\mathbf{R}}}^T_{1,n} \cdots \vec{\vec{\mathbf{R}}}^T_{1,2} = \vec{\vec{\mathbf{R}}}^T$ by the column vector $\left(\vec{\mathbf{x}} - \vec{\boldsymbol{\mu}} \right)$; hence, we have the following:

$$\begin{aligned}
y_{\substack{\text{upper-half} \\ \text{axis-aligned} \\ \text{ellipsoid}}} &= \sigma_{n+1} \sqrt{1 - \left(\vec{\mathbf{x}} - \vec{\boldsymbol{\mu}} \right)^T \vec{\vec{\mathbf{V}}}^{-1} \left(\vec{\mathbf{x}} - \vec{\boldsymbol{\mu}} \right)} \\
y_{\substack{\text{upper-half} \\ \text{rotated} \\ \text{ellipsoid}}} &= \sigma_{n+1} \sqrt{1 - \left(\vec{\vec{\mathbf{R}}}^T \left(\vec{\mathbf{x}} - \vec{\boldsymbol{\mu}} \right) \right)^T \vec{\vec{\mathbf{V}}}^{-1} \left(\vec{\vec{\mathbf{R}}}^T \left(\vec{\mathbf{x}} - \vec{\boldsymbol{\mu}} \right) \right)} \\
&= \sigma_{n+1} \sqrt{1 - \left(\vec{\mathbf{x}} - \vec{\boldsymbol{\mu}} \right)^T \left(\vec{\vec{\mathbf{R}}}^T \right)^T \vec{\vec{\mathbf{V}}}^{-1} \left(\vec{\vec{\mathbf{R}}}^T \right) \left(\vec{\mathbf{x}} - \vec{\boldsymbol{\mu}} \right)} \\
&= \sigma_{n+1} \sqrt{1 - \left(\vec{\mathbf{x}} - \vec{\boldsymbol{\mu}} \right)^T \left(\vec{\vec{\mathbf{R}}}^{-1} \right)^T \vec{\vec{\mathbf{V}}}^{-1} \left(\vec{\vec{\mathbf{R}}}^{-1} \right) \left(\vec{\mathbf{x}} - \vec{\boldsymbol{\mu}} \right)} \\
&= \sigma_{n+1} \sqrt{1 - \left(\vec{\mathbf{x}} - \vec{\boldsymbol{\mu}} \right)^T \left(\vec{\vec{\mathbf{R}}}^T \right)^{-1} \vec{\vec{\mathbf{V}}}^{-1} \left(\vec{\vec{\mathbf{R}}}^{-1} \right) \left(\vec{\mathbf{x}} - \vec{\boldsymbol{\mu}} \right)} \\
&= \sigma_{n+1} \sqrt{1 - \left(\vec{\mathbf{x}} - \vec{\boldsymbol{\mu}} \right)^T \left(\vec{\vec{\mathbf{R}}} \vec{\vec{\mathbf{V}}} \vec{\vec{\mathbf{R}}}^T \right)^{-1} \left(\vec{\mathbf{x}} - \vec{\boldsymbol{\mu}} \right)}
\end{aligned}$$

Example: Let x_1, x_2, y be variables aligned to the 3 axes of $\mathbb{R}^3$, then the equation for an axis-aligned ellipsoid centered at $\left(-1,-2,0\right)$ with radii $\left(2,3,1\right)$ along each of those 3 axes is the following:

$$\frac{\left(x_1-\left(-1\right)\right)^2}{\left(2\right)^2}+\frac{\left(x_2-\left(-2\right)\right)^2}{\left(3\right)^2}+\frac{\left(y-\left(0\right)\right)^2}{\left(1\right)^2}=1$$

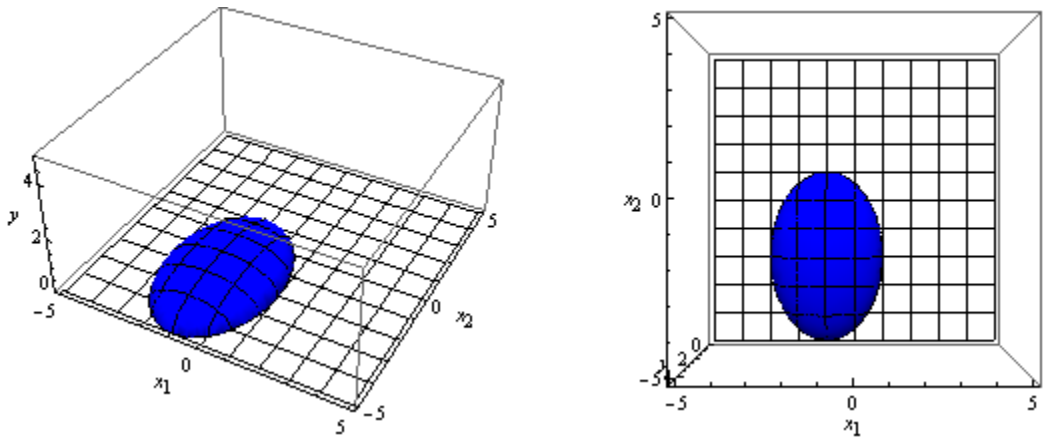
For the upper-half of this axis-aligned ellipsoid, we have the following:

$$y_{\substack{\text{upper-half}\\ \text{axis-aligned}\\ \text{ellipsoid}}}=\left(1\right)\sqrt{1-\left(\frac{\left(x_1-\left(-1\right)\right)^2}{\left(2\right)^2}+\frac{\left(x_2-\left(-2\right)\right)^2}{\left(3\right)^2}\right)}$$

and

$$\begin{aligned} y_{\substack{\text{upper-half}\\ \text{axis-aligned}\\ \text{ellipsoid}}} &= \left(1\right)\sqrt{1-\left(\begin{bmatrix}x_1\\x_2\end{bmatrix}-\begin{bmatrix}-1\\-2\end{bmatrix}\right)^T\left(\begin{bmatrix}\left(2\right)^2 & 0\\ 0 & \left(3\right)^2\end{bmatrix}\right)^{-1}\left(\begin{bmatrix}x_1\\x_2\end{bmatrix}-\begin{bmatrix}-1\\-2\end{bmatrix}\right)} \\ &= \left(1\right)\sqrt{1-\left(\begin{bmatrix}x_1-\left(-1\right)\\x_2-\left(-2\right)\end{bmatrix}\right)^T\left(\begin{bmatrix}\left(2\right)^2 & 0\\ 0 & \left(3\right)^2\end{bmatrix}\right)^{-1}\left(\begin{bmatrix}x_1-\left(-1\right)\\x_2-\left(-2\right)\end{bmatrix}\right)} \end{aligned}$$

which produce the following graph (perspective and top views, respectively):



If we rotate our upper-half ellipsoid around its center by $\theta = \frac{\pi}{6}$ radians in the $\vec{\mathbf{x}}_1$-$\vec{\mathbf{x}}_2$ hyperplane, then we have the following:

$$\begin{aligned}\vec{\vec{\mathbf{R}}}_{1,2} &= \begin{bmatrix} \cos\left(\theta\right) & \cos\left(\theta + \frac{\pi}{2}\right) \\ \sin\left(\theta\right) & \sin\left(\theta + \frac{\pi}{2}\right) \end{bmatrix} \\ &= \begin{bmatrix} \cos\left(\theta\right) & -\sin\left(\theta\right) \\ \sin\left(\theta\right) & \cos\left(\theta\right) \end{bmatrix} \\ &= \begin{bmatrix} \frac{\sqrt{3}}{2} & -\frac{1}{2} \\ \frac{1}{2} & \frac{\sqrt{3}}{2} \end{bmatrix}\end{aligned}$$

Moreover, $\vec{\vec{\mathbf{R}}} = \vec{\vec{\mathbf{R}}}_{1,2}$ and the rotated upper-half ellipsoid is the following:

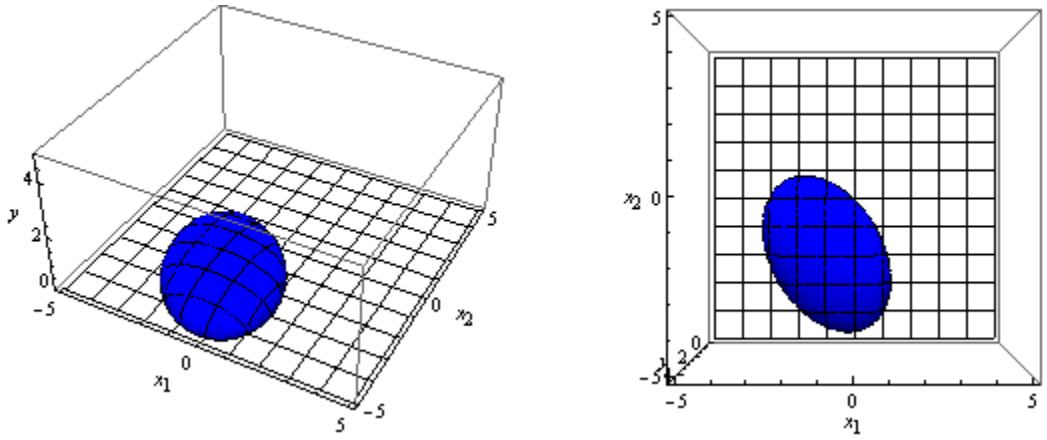
$$y_{\substack{\text{upper-half} \\ \text{rotated} \\ \text{ellipsoid}}} = \left(1\right)\sqrt{1 - \left(\begin{bmatrix} x_1 - \left(-1\right) \\ x_2 - \left(-2\right) \end{bmatrix}\right)^T \left(\begin{bmatrix} \frac{\sqrt{3}}{2} & -\frac{1}{2} \\ \frac{1}{2} & \frac{\sqrt{3}}{2} \end{bmatrix}\begin{bmatrix} \left(2\right)^2 & 0 \\ 0 & \left(3\right)^2 \end{bmatrix}\begin{bmatrix} \frac{\sqrt{3}}{2} & -\frac{1}{2} \\ \frac{1}{2} & \frac{\sqrt{3}}{2} \end{bmatrix}^T\right)^{-1} \left(\begin{bmatrix} x_1 - \left(-1\right) \\ x_2 - \left(-2\right) \end{bmatrix}\right)}$$

and

$$y_{\substack{\text{rotated}\\ \text{upper-half}\\ \text{ellipsoid}}} = (1)\sqrt{1-\left(\frac{\cos\left(\frac{\pi}{6}\right)\left(x_1-(-1)\right)^2+\sin\left(\frac{\pi}{6}\right)\left(x_2-(-2)\right)^2}{(2)^2}+\frac{-\sin\left(\frac{\pi}{6}\right)\left(x_1-(-1)\right)^2+\cos\left(\frac{\pi}{6}\right)\left(x_2-(-2)\right)^2}{(3)^2}\right)}$$

$$= (1)\sqrt{1-\left(\frac{\frac{\sqrt{3}}{2}\left(x_1-(-1)\right)^2+\frac{1}{2}\left(x_2-(-2)\right)^2}{(2)^2}+\frac{-\frac{1}{2}\left(x_1-(-1)\right)^2+\frac{\sqrt{3}}{2}\left(x_2-(-2)\right)^2}{(3)^2}\right)}$$

which produces the following graph (perspective and top views, respectively):



Similarly, we can also rotate the first approximation to the Taylor series expansion of the upper-half of our axis-aligned ellipsoid; hence, we have the following:

$$y_{\substack{\text{first approximation}\\ \text{to upper-half axis-}\\ \text{aligned ellipsoid}}} = \sigma_{n+1}\exp\left(-\frac{1}{2}\left(\vec{\mathbf{x}}-\vec{\boldsymbol{\mu}}\right)^T\vec{\vec{\mathbf{V}}}^{-1}\left(\vec{\mathbf{x}}-\vec{\boldsymbol{\mu}}\right)\right)$$

$$y_{\substack{\text{first approximation}\\ \text{to upper-half}\\ \text{rotated ellipsoid}}} = \sigma_{n+1}\exp\left(-\frac{1}{2}\left(\vec{\mathbf{x}}-\vec{\boldsymbol{\mu}}\right)^T\left(\vec{\vec{\mathbf{R}}}\vec{\vec{\mathbf{V}}}\vec{\vec{\mathbf{R}}}^T\right)^{-1}\left(\vec{\mathbf{x}}-\vec{\boldsymbol{\mu}}\right)\right)$$

Moreover, we can also rotate the normalized first approximation to the Taylor series expansion of the upper-half of our axis-aligned ellipsoid; hence, we also have the following:

$$y_{\substack{\text{normalized}\\ \text{first approximation}\\ \text{to upper-half axis-}\\ \text{aligned ellipsoid}}} = \frac{1}{\left(2\pi\right)^{\frac{n}{2}}\left|\vec{\vec{\mathbf{V}}}\right|^{\frac{1}{2}}}\exp\left(-\frac{1}{2}\left(\vec{\mathbf{x}}-\vec{\boldsymbol{\mu}}\right)^{T}\vec{\vec{\mathbf{V}}}^{-1}\left(\vec{\mathbf{x}}-\vec{\boldsymbol{\mu}}\right)\right)$$

$$y_{\substack{\text{normalized}\\ \text{first approximation}\\ \text{to upper-half}\\ \text{rotated ellipsoid}}} = \frac{1}{\left(2\pi\right)^{\frac{n}{2}}\left|\left(\vec{\vec{\mathbf{R}}}\vec{\vec{\mathbf{V}}}\vec{\vec{\mathbf{R}}}^{T}\right)\right|^{\frac{1}{2}}}\exp\left(-\frac{1}{2}\left(\vec{\mathbf{x}}-\vec{\boldsymbol{\mu}}\right)^{T}\left(\vec{\vec{\mathbf{R}}}\vec{\vec{\mathbf{V}}}\vec{\vec{\mathbf{R}}}^{T}\right)^{-1}\left(\vec{\mathbf{x}}-\vec{\boldsymbol{\mu}}\right)\right)$$

The full covariance matrix $\vec{\vec{\boldsymbol{\Sigma}}}$ (i.e., the covariance matrix that is not necessarily axis-aligned) can be recovered from the following singular value decomposition:

$$\vec{\vec{\mathbf{B}}}\vec{\vec{\mathbf{V}}}_{\text{ordered}}\vec{\vec{\mathbf{B}}}^{T} = \vec{\vec{\boldsymbol{\Sigma}}}$$

where $\vec{\vec{\mathbf{B}}}$ (for basis) is an orthogonal matrix such that $\left|\vec{\vec{\mathbf{B}}}\right| = \pm 1$ and $\vec{\vec{\mathbf{V}}}_{\text{ordered}}$ is $\vec{\vec{\mathbf{V}}}$ with the squared radii (i.e., the variances) ordered along the diagonal. Thus, we need to find $\vec{\vec{\mathbf{B}}}$ to recover $\vec{\vec{\boldsymbol{\Sigma}}}$.

In order to find $\vec{\vec{\mathbf{B}}}$, we now turn to row and column swapping elementary matrices. Let $\vec{\vec{\mathbf{E}}}_{\substack{\text{row}\\ \text{swap}\\ w}}$ and $\vec{\vec{\mathbf{E}}}_{\substack{\text{column}\\ \text{swap}\\ w}}$ denote the w-th row and column swapping elementary matrix, respectively. Row and column swapping elementary matrices can be used to order values along the diagonal of a matrix. Since we need to order $\vec{\vec{\mathbf{V}}}$, then we will use elementary matrices to accomplish that task; hence, we have the following:

$$\vec{\vec{\mathbf{V}}}_{\text{ordered}} = \underbrace{\vec{\vec{\mathbf{E}}}_{\substack{\text{row}\\ \text{swap}\\ q}}\cdots\vec{\vec{\mathbf{E}}}_{\substack{\text{row}\\ \text{swap}\\ 1}}}_{q}\ \vec{\vec{\mathbf{V}}}\ \underbrace{\vec{\vec{\mathbf{E}}}_{\substack{\text{column}\\ \text{swap}\\ 1}}\cdots\vec{\vec{\mathbf{E}}}_{\substack{\text{column}\\ \text{swap}\\ q}}}_{q}$$

where $\vec{\vec{\mathbf{E}}}_{\substack{\text{row}\\ \text{swap}\\ w}} = \vec{\vec{\mathbf{E}}}_{\substack{\text{column}\\ \text{swap}\\ w}}$ for $w = 1,\ldots,q$.

Example: To order the squared radii along the diagonal of the following matrix:

$$\begin{bmatrix} (4)^2 & 0 & 0 & 0 \\ 0 & (2)^2 & 0 & 0 \\ 0 & 0 & (1)^2 & 0 \\ 0 & 0 & 0 & (3)^2 \end{bmatrix}$$

We use the following elementary matrices:

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix} \text{ and } \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \end{bmatrix}$$

such that we have the following:

$$\left(\underbrace{\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}}_{\substack{\text{swap row 3} \\ \text{and row 4}}} \left(\underbrace{\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}}_{\substack{\text{swap row 2} \\ \text{and row 4}}} \begin{bmatrix} (4)^2 & 0 & 0 & 0 \\ 0 & (2)^2 & 0 & 0 \\ 0 & 0 & (1)^2 & 0 \\ 0 & 0 & 0 & (3)^2 \end{bmatrix} \underbrace{\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}}_{\substack{\text{swap column 2} \\ \text{and column 4}}} \right) \underbrace{\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}}_{\substack{\text{swap column 3} \\ \text{and column 4}}} \right)$$

$$= \underbrace{\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}}_{\substack{\text{swap row 3} \\ \text{and row 4}}} \begin{bmatrix} (4)^2 & 0 & 0 & 0 \\ 0 & (3)^2 & 0 & 0 \\ 0 & 0 & (1)^2 & 0 \\ 0 & 0 & 0 & (2)^2 \end{bmatrix} \underbrace{\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}}_{\substack{\text{swap column 3} \\ \text{and column 4}}}$$

$$= \begin{bmatrix} (4)^2 & 0 & 0 & 0 \\ 0 & (3)^2 & 0 & 0 \\ 0 & 0 & (2)^2 & 0 \\ 0 & 0 & 0 & (1)^2 \end{bmatrix}$$

Row swapping and column swapping elementary matrices also have the following properties:

1) $\underset{\substack{\text{row}\\\text{swap}\\w}}{\vec{\vec{\mathbf{E}}}}$ and $\underset{\substack{\text{column}\\\text{swap}\\w}}{\vec{\vec{\mathbf{E}}}}$ are orthogonal matrices with components of value 0 or 1 only

2) $\underset{\substack{\text{row}\\\text{swap}\\w}}{\vec{\vec{\mathbf{E}}}} = \underset{\substack{\text{row}\\\text{swap}\\w}}{\vec{\vec{\mathbf{E}}}^{T}} = \underset{\substack{\text{row}\\\text{swap}\\w}}{\vec{\vec{\mathbf{E}}}^{-1}}$ and $\underset{\substack{\text{column}\\\text{swap}\\w}}{\vec{\vec{\mathbf{E}}}} = \underset{\substack{\text{column}\\\text{swap}\\w}}{\vec{\vec{\mathbf{E}}}^{T}} = \underset{\substack{\text{column}\\\text{swap}\\w}}{\vec{\vec{\mathbf{E}}}^{-1}}$

Since $\vec{\vec{\mathbf{V}}}_{\text{ordered}} = \underbrace{\underset{\substack{\text{row}\\\text{swap}\\q}}{\vec{\vec{\mathbf{E}}}} \cdots \underset{\substack{\text{row}\\\text{swap}\\1}}{\vec{\vec{\mathbf{E}}}}}_{q} \vec{\vec{\mathbf{V}}} \underbrace{\underset{\substack{\text{column}\\\text{swap}\\1}}{\vec{\vec{\mathbf{E}}}} \cdots \underset{\substack{\text{column}\\\text{swap}\\q}}{\vec{\vec{\mathbf{E}}}}}_{q}$, $\underset{\substack{\text{row}\\\text{swap}\\w}}{\vec{\vec{\mathbf{E}}}^{-1}} = \underset{\substack{\text{row}\\\text{swap}\\w}}{\vec{\vec{\mathbf{E}}}}$, and $\underset{\substack{\text{column}\\\text{swap}\\w}}{\vec{\vec{\mathbf{E}}}^{-1}} = \underset{\substack{\text{column}\\\text{swap}\\w}}{\vec{\vec{\mathbf{E}}}}$,

then we have the following:

$$\underbrace{\underset{\substack{\text{row}\\\text{swap}\\q}}{\vec{\vec{\mathbf{E}}}} \cdots \underset{\substack{\text{row}\\\text{swap}\\1}}{\vec{\vec{\mathbf{E}}}}}_{q} \vec{\vec{\mathbf{V}}} \underbrace{\underset{\substack{\text{column}\\\text{swap}\\1}}{\vec{\vec{\mathbf{E}}}} \cdots \underset{\substack{\text{column}\\\text{swap}\\q}}{\vec{\vec{\mathbf{E}}}}}_{q} = \vec{\vec{\mathbf{V}}}_{\text{ordered}}$$

$$\underbrace{\underset{\substack{\text{row}\\\text{swap}\\1}}{\vec{\vec{\mathbf{E}}}^{-1}} \cdots \underset{\substack{\text{row}\\\text{swap}\\q}}{\vec{\vec{\mathbf{E}}}^{-1}}}_{q} \underbrace{\underset{\substack{\text{row}\\\text{swap}\\q}}{\vec{\vec{\mathbf{E}}}} \cdots \underset{\substack{\text{row}\\\text{swap}\\1}}{\vec{\vec{\mathbf{E}}}}}_{q} \vec{\vec{\mathbf{V}}} \underbrace{\underset{\substack{\text{column}\\\text{swap}\\1}}{\vec{\vec{\mathbf{E}}}} \cdots \underset{\substack{\text{column}\\\text{swap}\\q}}{\vec{\vec{\mathbf{E}}}}}_{q} = \underbrace{\underset{\substack{\text{row}\\\text{swap}\\1}}{\vec{\vec{\mathbf{E}}}^{-1}} \cdots \underset{\substack{\text{row}\\\text{swap}\\q}}{\vec{\vec{\mathbf{E}}}^{-1}}}_{q} \vec{\vec{\mathbf{V}}}_{\text{ordered}}$$

$$\vec{\vec{\mathbf{V}}} \underbrace{\underset{\substack{\text{column}\\\text{swap}\\1}}{\vec{\vec{\mathbf{E}}}} \cdots \underset{\substack{\text{column}\\\text{swap}\\q}}{\vec{\vec{\mathbf{E}}}}}_{q} \underbrace{\underset{\substack{\text{column}\\\text{swap}\\q}}{\vec{\vec{\mathbf{E}}}^{-1}} \cdots \underset{\substack{\text{column}\\\text{swap}\\1}}{\vec{\vec{\mathbf{E}}}^{-1}}}_{q} = \underbrace{\underset{\substack{\text{row}\\\text{swap}\\1}}{\vec{\vec{\mathbf{E}}}^{-1}} \cdots \underset{\substack{\text{row}\\\text{swap}\\q}}{\vec{\vec{\mathbf{E}}}^{-1}}}_{q} \vec{\vec{\mathbf{V}}}_{\text{ordered}} \underbrace{\underset{\substack{\text{column}\\\text{swap}\\q}}{\vec{\vec{\mathbf{E}}}^{-1}} \cdots \underset{\substack{\text{column}\\\text{swap}\\1}}{\vec{\vec{\mathbf{E}}}^{-1}}}_{q}$$

$$\vec{\vec{\mathbf{V}}} = \underbrace{\underset{\substack{\text{row}\\\text{swap}\\1}}{\vec{\vec{\mathbf{E}}}^{-1}} \cdots \underset{\substack{\text{row}\\\text{swap}\\q}}{\vec{\vec{\mathbf{E}}}^{-1}}}_{q} \vec{\vec{\mathbf{V}}}_{\text{ordered}} \underbrace{\underset{\substack{\text{column}\\\text{swap}\\q}}{\vec{\vec{\mathbf{E}}}^{-1}} \cdots \underset{\substack{\text{column}\\\text{swap}\\1}}{\vec{\vec{\mathbf{E}}}^{-1}}}_{q}$$

$$\vec{\vec{\mathbf{V}}} = \underbrace{\underset{\substack{\text{row}\\\text{swap}\\1}}{\vec{\vec{\mathbf{E}}}} \cdots \underset{\substack{\text{row}\\\text{swap}\\q}}{\vec{\vec{\mathbf{E}}}}}_{q} \vec{\vec{\mathbf{V}}}_{\text{ordered}} \underbrace{\underset{\substack{\text{column}\\\text{swap}\\q}}{\vec{\vec{\mathbf{E}}}} \cdots \underset{\substack{\text{column}\\\text{swap}\\1}}{\vec{\vec{\mathbf{E}}}}}_{q}$$

Moreover, since $\vec{\vec{\mathbf{\Sigma}}}_{\text{axis-aligned}} = \vec{\vec{\mathbf{V}}}$, then we have the following:

$$\begin{aligned}\vec{\vec{\mathbf{\Sigma}}}_{\text{axis-aligned}} &= \vec{\vec{\mathbf{V}}} \\ &= \underbrace{\underset{\substack{\text{row}\\\text{swap}\\1}}{\vec{\vec{\mathbf{E}}}} \cdots \underset{\substack{\text{row}\\\text{swap}\\q}}{\vec{\vec{\mathbf{E}}}}}_{q} \vec{\vec{\mathbf{V}}}_{\text{ordered}} \underbrace{\underset{\substack{\text{column}\\\text{swap}\\q}}{\vec{\vec{\mathbf{E}}}} \cdots \underset{\substack{\text{column}\\\text{swap}\\1}}{\vec{\vec{\mathbf{E}}}}}_{q}\end{aligned}$$

Since we are reordering the squared radii (the variances) along the diagonal of $\overset{=}{\mathbf{V}}$, then $\underset{\substack{\text{row}\\\text{swap}\\w}}{\overset{=}{\mathbf{E}}} = \underset{\substack{\text{column}\\\text{swap}\\w}}{\overset{=}{\mathbf{E}}}$ for $w = 1,\ldots,q$, so we will drop the "row swap" and "column swap" labels and just write $\overset{=}{\mathbf{E}}_w$ such that we have the following:

$$\overset{=}{\mathbf{\Sigma}}_{\text{axis-aligned}} = \underbrace{\overset{=}{\mathbf{E}}_1 \cdots \overset{=}{\mathbf{E}}_q}_{q} \overset{=}{\mathbf{V}}_{\text{ordered}} \underbrace{\overset{=}{\mathbf{E}}_q \cdots \overset{=}{\mathbf{E}}_1}_{q}$$

Moreover, since $\overset{=}{\mathbf{E}}_w = \overset{=}{\mathbf{E}}_w^{\,T}$, then we have the following:

$$\begin{aligned}
\overset{=}{\mathbf{\Sigma}}_{\text{axis-aligned}} &= \underbrace{\overset{=}{\mathbf{E}}_1 \cdots \overset{=}{\mathbf{E}}_q}_{q} \overset{=}{\mathbf{V}}_{\text{ordered}} \underbrace{\overset{=}{\mathbf{E}}_q \cdots \overset{=}{\mathbf{E}}_1}_{q} \\
&= \left(\overset{=}{\mathbf{E}}_1 \cdots \overset{=}{\mathbf{E}}_q\right) \overset{=}{\mathbf{V}}_{\text{ordered}} \left(\left(\overset{=}{\mathbf{E}}_q \cdots \overset{=}{\mathbf{E}}_1\right)^T\right)^T \\
&= \left(\overset{=}{\mathbf{E}}_1 \cdots \overset{=}{\mathbf{E}}_q\right) \overset{=}{\mathbf{V}}_{\text{ordered}} \left(\overset{=}{\mathbf{E}}_1^T \cdots \overset{=}{\mathbf{E}}_q^T\right)^T \\
&= \left(\overset{=}{\mathbf{E}}_1 \cdots \overset{=}{\mathbf{E}}_q\right) \overset{=}{\mathbf{V}}_{\text{ordered}} \left(\overset{=}{\mathbf{E}}_1 \cdots \overset{=}{\mathbf{E}}_q\right)^T
\end{aligned}$$

If we let $\left(\overset{=}{\mathbf{E}}_1 \cdots \overset{=}{\mathbf{E}}_q\right) = \overset{=}{\mathbf{B}}_{\text{axis-aligned}}$, then $\overset{=}{\mathbf{B}}_{\text{axis-aligned}}$ is an orthogonal matrix with $\left|\overset{=}{\mathbf{B}}_{\text{axis-aligned}}\right| = \pm 1$ and we have a singular value decomposition $\overset{=}{\mathbf{\Sigma}}_{\text{axis-aligned}} = \overset{=}{\mathbf{B}}_{\text{axis-aligned}} \overset{=}{\mathbf{V}}_{\text{ordered}} \overset{=}{\mathbf{B}}_{\text{axis-aligned}}^{\,T}$ where the diagonal of $\overset{=}{\mathbf{V}}_{\text{ordered}}$ are the eigenvalues (i.e., the squared radii or variances) and $\overset{=}{\mathbf{B}}_{\text{axis-aligned}}$ are the corresponding eigenvectors (i.e., the vectors along which the radii are aligned).

Since we can also "rotate" $\overset{=}{\mathbf{V}}$ using $\overset{=}{\mathbf{R}}\overset{=}{\mathbf{V}}\overset{=}{\mathbf{R}}^T$ where $\overset{=}{\mathbf{R}}$ is as previously defined, then we also have the following:

$$\begin{aligned}
\overset{=}{\mathbf{R}}\overset{=}{\mathbf{V}}\overset{=}{\mathbf{R}}^T &= \overset{=}{\mathbf{R}}\overset{=}{\mathbf{\Sigma}}_{\text{axis-aligned}}\overset{=}{\mathbf{R}}^T \\
&= \overset{=}{\mathbf{R}}\left(\overset{=}{\mathbf{E}}_1 \cdots \overset{=}{\mathbf{E}}_q\right) \overset{=}{\mathbf{V}}_{\text{ordered}} \left(\overset{=}{\mathbf{E}}_1 \cdots \overset{=}{\mathbf{E}}_q\right)^T \overset{=}{\mathbf{R}}^T \\
&= \left(\overset{=}{\mathbf{R}}\left(\overset{=}{\mathbf{E}}_1 \cdots \overset{=}{\mathbf{E}}_q\right)\right) \overset{=}{\mathbf{V}}_{\text{ordered}} \left(\overset{=}{\mathbf{R}}\left(\overset{=}{\mathbf{E}}_1 \cdots \overset{=}{\mathbf{E}}_q\right)\right)^T
\end{aligned}$$

Since the singular value decomposition of $\vec{\vec{\Sigma}}$ is $\vec{\vec{\mathbf{B}}}\vec{\vec{\mathbf{V}}}_{\text{ordered}}\vec{\vec{\mathbf{B}}}^T$ where $\vec{\vec{\mathbf{B}}}$ is not necessarily axis-aligned, then we have that $\vec{\vec{\mathbf{B}}} = \vec{\vec{\mathbf{R}}}\left(\vec{\vec{\mathbf{E}}}_1 \cdots \vec{\vec{\mathbf{E}}}_q\right)$. Moreover, since $\vec{\vec{\mathbf{R}}} = \vec{\vec{\mathbf{R}}}_{1,2} \cdots \vec{\vec{\mathbf{R}}}_{1,n}\vec{\vec{\mathbf{R}}}_{2,3} \cdots \vec{\vec{\mathbf{R}}}_{2,n}\vec{\vec{\mathbf{R}}}_{3,4} \cdots \vec{\vec{\mathbf{R}}}_{3,n} \cdots \vec{\vec{\mathbf{R}}}_{n-1,n}$, the we have the following:

$$\vec{\vec{\mathbf{B}}} = \left(\vec{\vec{\mathbf{R}}}_{1,2} \cdots \vec{\vec{\mathbf{R}}}_{1,n}\vec{\vec{\mathbf{R}}}_{2,3} \cdots \vec{\vec{\mathbf{R}}}_{2,n}\vec{\vec{\mathbf{R}}}_{3,4} \cdots \vec{\vec{\mathbf{R}}}_{3,n} \cdots \vec{\vec{\mathbf{R}}}_{n-1,n}\right)\left(\vec{\vec{\mathbf{E}}}_1 \cdots \vec{\vec{\mathbf{E}}}_q\right)$$

Thus, the eigenvectors associated with the covariance matrix $\vec{\vec{\Sigma}}$ are a result of the "rotation" and ordering of the eigenvalues (i.e., the ordered squared radii or the variances). Hence, the covariance matrix $\vec{\vec{\Sigma}} = \vec{\vec{\mathbf{B}}}\vec{\vec{\mathbf{V}}}_{\text{ordered}}\vec{\vec{\mathbf{B}}}^T = \vec{\vec{\mathbf{R}}}\vec{\vec{\mathbf{V}}}\vec{\vec{\mathbf{R}}}^T$.

Therefore, the $n+1$ dimensional upper-half ellipsoid that is rotated in a n-dimensional hyperplane about its center is the following:

$$\begin{aligned} y_{\substack{\text{upper-half}\\ \text{rotated}\\ \text{ellipsoid}}} &= \sigma_{n+1}\sqrt{1 - \left(\vec{\mathbf{x}} - \vec{\boldsymbol{\mu}}\right)^T \left(\vec{\vec{\mathbf{R}}}\vec{\vec{\mathbf{V}}}\vec{\vec{\mathbf{R}}}^T\right)^{-1} \left(\vec{\mathbf{x}} - \vec{\boldsymbol{\mu}}\right)} \\ &= \sigma_{n+1}\sqrt{1 - \left(\vec{\mathbf{x}} - \vec{\boldsymbol{\mu}}\right)^T \vec{\vec{\boldsymbol{\Sigma}}}^{-1} \left(\vec{\mathbf{x}} - \vec{\boldsymbol{\mu}}\right)} \end{aligned}$$

The first approximation to the Taylor series expansion of the upper-half of our rotated ellipsoid above is the following:

$$\begin{aligned} y_{\substack{\text{first approximation}\\ \text{to upper-half}\\ \text{rotated ellipsoid}}} &= \sigma_{n+1} \exp\left(-\frac{1}{2}\left(\vec{\mathbf{x}} - \vec{\boldsymbol{\mu}}\right)^T \left(\vec{\vec{\mathbf{R}}}\vec{\vec{\mathbf{V}}}\vec{\vec{\mathbf{R}}}^T\right)^{-1} \left(\vec{\mathbf{x}} - \vec{\boldsymbol{\mu}}\right)\right) \\ &= \sigma_{n+1} \exp\left(-\frac{1}{2}\left(\vec{\mathbf{x}} - \vec{\boldsymbol{\mu}}\right)^T \vec{\vec{\boldsymbol{\Sigma}}}^{-1} \left(\vec{\mathbf{x}} - \vec{\boldsymbol{\mu}}\right)\right) \end{aligned}$$

The normalized first approximation to the Taylor series expansion of the upper-half of our rotated ellipsoid above is the following:

$$y_{\substack{\text{normalized}\\ \text{first approximation}\\ \text{to upper-half}\\ \text{rotated ellipsoid}}} = \frac{1}{\left(2\pi\right)^{\frac{n}{2}}\left|\left(\vec{\vec{\mathbf{R}}}\vec{\vec{\mathbf{V}}}\vec{\vec{\mathbf{R}}}^{T}\right)\right|^{\frac{1}{2}}}\exp\left(-\frac{1}{2}\left(\vec{\mathbf{x}}-\vec{\boldsymbol{\mu}}\right)^{T}\left(\vec{\vec{\mathbf{R}}}\vec{\vec{\mathbf{V}}}\vec{\vec{\mathbf{R}}}^{T}\right)^{-1}\left(\vec{\mathbf{x}}-\vec{\boldsymbol{\mu}}\right)\right)$$
$$= \frac{1}{\left(2\pi\right)^{\frac{n}{2}}\left|\vec{\vec{\boldsymbol{\Sigma}}}\right|^{\frac{1}{2}}}\exp\left(-\frac{1}{2}\left(\vec{\mathbf{x}}-\vec{\boldsymbol{\mu}}\right)^{T}\vec{\vec{\boldsymbol{\Sigma}}}^{-1}\left(\vec{\mathbf{x}}-\vec{\boldsymbol{\mu}}\right)\right)$$

Therefore, the normalized first approximation to the Taylor series expansion of the upper-half of the $n+1$ dimensional ellipsoid centered at $\left(\mu_1,\ldots,\mu_n,0\right)$ and rotated in a hyperplane restricted to the first n dimensions is the probability density function of the multivariate Normal distribution.

B. COROLLARY: THE PROBABILITY DENSITY FUNCTION OF THE MULTIVARIATE NORMAL DISTRIBUTION CENTERED AT $\vec{\boldsymbol{\mu}}=\left(\mu_1,\ldots,\mu_n\right)$ WITH NON-SINGULAR COVARIANCE $\vec{\vec{\boldsymbol{\Sigma}}}$ IS BOUNDED BELOW BY THE SECOND OR HIGHER APPROXIMATION TO THE TAYLOR SERIES EXPANSION OF THE UPPER-HALF OF THE $n+1$ DIMENSIONAL ELLIPSOID WITH IDENTICAL COVARIANCE $\vec{\vec{\boldsymbol{\Sigma}}}$ CENTERED AT $\left(\mu_1,\ldots,\mu_n,0\right)$ AND MULTIPLIED BY THE SCALAR $1\Big/\left(\left(2\pi\right)^{n/2}\left|\vec{\vec{\boldsymbol{\Sigma}}}\right|^{1/2}\right)$.

The probability density function of the multivariate Normal distribution is the following:

$$y_{\substack{\text{pdf of the}\\ \text{multivariate}\\ \text{normal}}} = \frac{1}{\left(2\pi\right)^{\frac{n}{2}}\left|\vec{\vec{\boldsymbol{\Sigma}}}\right|^{\frac{1}{2}}}\exp\left(-\frac{1}{2}\left(\vec{\mathbf{x}}-\vec{\boldsymbol{\mu}}\right)^{T}\vec{\vec{\boldsymbol{\Sigma}}}^{-1}\left(\vec{\mathbf{x}}-\vec{\boldsymbol{\mu}}\right)\right)$$

The maximum value of this function occurs when $\vec{\mathbf{x}}=\vec{\boldsymbol{\mu}}$. When $\vec{\mathbf{x}}=\vec{\boldsymbol{\mu}}$, then $\vec{\mathbf{x}}-\vec{\boldsymbol{\mu}}=\vec{\mathbf{0}}$ and we have the following:

$$
\begin{aligned}
y_{\substack{\text{pdf of the}\\ \text{multivariate}\\ \text{normal}}} &= \frac{1}{\left(2\pi\right)^{\frac{n}{2}}\left|\vec{\vec{\Sigma}}\right|^{\frac{1}{2}}}\exp\left(-\frac{1}{2}\left(\vec{\mathbf{x}}-\vec{\boldsymbol{\mu}}\right)^{T}\vec{\vec{\Sigma}}^{-1}\left(\vec{\mathbf{x}}-\vec{\boldsymbol{\mu}}\right)\right)\\
y_{\substack{\text{pdf of the}\\ \text{multivariate}\\ \text{normal with}\\ \vec{\mathbf{x}}-\vec{\boldsymbol{\mu}}=\vec{\mathbf{0}}}} &= \frac{1}{\left(2\pi\right)^{\frac{n}{2}}\left|\vec{\vec{\Sigma}}\right|^{\frac{1}{2}}}\exp\left(-\frac{1}{2}\left(\vec{\mathbf{0}}\right)^{T}\vec{\vec{\Sigma}}^{-1}\left(\vec{\mathbf{0}}\right)\right)\\
&= \frac{1}{\left(2\pi\right)^{\frac{n}{2}}\left|\vec{\vec{\Sigma}}\right|^{\frac{1}{2}}}\exp\left(0\right)\\
&= \frac{1}{\left(2\pi\right)^{\frac{n}{2}}\left|\vec{\vec{\Sigma}}\right|^{\frac{1}{2}}}
\end{aligned}
$$

Hence, the maximum value of the probability density function of the multivariate Normal distribution is $1\Big/\left(\left(2\pi\right)^{n/2}\left|\vec{\vec{\Sigma}}\right|^{1/2}\right)$.

Let $\sigma_{n+1}=1\Big/\left(\left(2\pi\right)^{n/2}\left|\vec{\vec{\Sigma}}\right|^{1/2}\right)$ where $\vec{\vec{\Sigma}}$ is the identical covariance of the probability density function of the multivariate Normal distribution above. Then we have the following scaled Taylor series expansion of the $n+1$ dimensional rotated ellipsoid:

$$
\begin{aligned}
y_{\substack{\text{upper-half}\\ \text{axis-aligned}\\ \text{ellipsoid}}} &= \sigma_{n+1}\exp\left(-\sum_{i=1}^{\infty}\frac{\left(\sum_{j=1}^{n}\frac{\left(x_{j}-\mu_{j}\right)^{2}}{\sigma_{j}^{2}}\right)^{i}}{2i}\right)\\
y_{\substack{\text{upper-half}\\ \text{rotated}\\ \text{ellipsoid}}} &= \sigma_{n+1}\exp\left(-\sum_{i=1}^{\infty}\frac{\left(\left(\vec{\mathbf{x}}-\vec{\boldsymbol{\mu}}\right)^{T}\vec{\vec{\Sigma}}^{-1}\left(\vec{\mathbf{x}}-\vec{\boldsymbol{\mu}}\right)\right)^{i}}{2i}\right)\\
y_{\substack{\text{scaled upper-half}\\ \text{rotated ellipsoid}}} &= \frac{1}{\left(2\pi\right)^{\frac{n}{2}}\left|\vec{\vec{\Sigma}}\right|^{\frac{1}{2}}}\exp\left(-\sum_{i=1}^{\infty}\frac{\left(\left(\vec{\mathbf{x}}-\vec{\boldsymbol{\mu}}\right)^{T}\vec{\vec{\Sigma}}^{-1}\left(\vec{\mathbf{x}}-\vec{\boldsymbol{\mu}}\right)\right)^{i}}{2i}\right)
\end{aligned}
$$

Similarly, the maximum value of this function occurs when $\vec{\mathbf{x}}=\vec{\boldsymbol{\mu}}$. When $\vec{\mathbf{x}}=\vec{\boldsymbol{\mu}}$, then $\vec{\mathbf{x}}-\vec{\boldsymbol{\mu}}=\vec{\mathbf{0}}$ and we also have the following:

$$
\begin{aligned}
y_{\substack{\text{scaled upper-half}\\ \text{rotated ellipsoid}}} &= \frac{1}{\left(2\pi\right)^{\frac{n}{2}}\left|\vec{\vec{\mathbf{\Sigma}}}\right|^{\frac{1}{2}}}\exp\left(-\sum_{i=1}^{\infty}\frac{\left(\left(\vec{\mathbf{x}}-\vec{\boldsymbol{\mu}}\right)^{T}\vec{\vec{\mathbf{\Sigma}}}^{-1}\left(\vec{\mathbf{x}}-\vec{\boldsymbol{\mu}}\right)\right)^{i}}{2i}\right)\\
y_{\substack{\text{scaled upper-half}\\ \text{rotated ellipsoid}\\ \text{with } \vec{\mathbf{x}}-\vec{\boldsymbol{\mu}}=\vec{\mathbf{0}}}} &= \frac{1}{\left(2\pi\right)^{\frac{n}{2}}\left|\vec{\vec{\mathbf{\Sigma}}}\right|^{\frac{1}{2}}}\exp\left(-\sum_{i=1}^{\infty}\frac{\left(\left(\vec{\mathbf{0}}\right)^{T}\vec{\vec{\mathbf{\Sigma}}}^{-1}\left(\vec{\mathbf{0}}\right)\right)^{i}}{2i}\right)\\
&= \frac{1}{\left(2\pi\right)^{\frac{n}{2}}\left|\vec{\vec{\mathbf{\Sigma}}}\right|^{\frac{1}{2}}}\exp\left(-\sum_{i=1}^{\infty}\frac{\left(0\right)^{i}}{2i}\right)\\
&= \frac{1}{\left(2\pi\right)^{\frac{n}{2}}\left|\vec{\vec{\mathbf{\Sigma}}}\right|^{\frac{1}{2}}}\exp\left(0\right)\\
&= \frac{1}{\left(2\pi\right)^{\frac{n}{2}}\left|\vec{\vec{\mathbf{\Sigma}}}\right|^{\frac{1}{2}}}
\end{aligned}
$$

Hence, at $\vec{\mathbf{x}}=\vec{\boldsymbol{\mu}}$, we have $y_{\substack{\text{pdf of the}\\ \text{multivariate}\\ \text{normal with}\\ \vec{\mathbf{x}}-\vec{\boldsymbol{\mu}}=\vec{\mathbf{0}}}} = y_{\substack{\text{scaled upper-half}\\ \text{rotated ellipsoid}\\ \text{with } \vec{\mathbf{x}}-\vec{\boldsymbol{\mu}}=\vec{\mathbf{0}}}}$ which implies that we also have

$y_{\substack{\text{pdf of the}\\ \text{multivariate}\\ \text{normal with}\\ \vec{\mathbf{x}}-\vec{\boldsymbol{\mu}}=\vec{\mathbf{0}}}} \geq y_{\substack{\text{scaled upper-half}\\ \text{rotated ellipsoid}\\ \text{with } \vec{\mathbf{x}}-\vec{\boldsymbol{\mu}}=\vec{\mathbf{0}}}}$.

Since $\vec{\vec{\mathbf{\Sigma}}}$ is non-singular, then $\vec{\vec{\mathbf{\Sigma}}}^{-1}$ exists and, from the details of the previous theorem, $\vec{\vec{\mathbf{\Sigma}}}=\vec{\vec{\mathbf{R}}}\vec{\vec{\mathbf{V}}}\vec{\vec{\mathbf{R}}}^{T}$ where $\vec{\vec{\mathbf{R}}}$ is a composite of rotation matrices and $\vec{\vec{\mathbf{V}}}$ is a diagonal matrix with entries corresponding to the squared radii of the axis-aligned ellipsoid (i.e., the radii of the ellipsoid prior to its current rotated state). Since all these squared radii are positive, then $\left|\vec{\vec{\mathbf{V}}}\right|>0$. Since $\left|\vec{\vec{\mathbf{X}}}\vec{\vec{\mathbf{Y}}}\right|=\left|\vec{\vec{\mathbf{X}}}\right|\left|\vec{\vec{\mathbf{Y}}}\right|$, $\left|\vec{\vec{\mathbf{X}}}^{T}\right|=\left|\vec{\vec{\mathbf{X}}}\right|$, $\left|\vec{\vec{\mathbf{V}}}\right|>0$, and $\left|\vec{\vec{\mathbf{R}}}\right|=\pm 1$, then we have the following:

$$\begin{aligned}\left|\vec{\vec{\boldsymbol{\Sigma}}}\right| &= \left|\vec{\vec{\mathbf{R}}}\vec{\vec{\mathbf{V}}}\vec{\vec{\mathbf{R}}}^{T}\right| \\ &= \left|\vec{\vec{\mathbf{R}}}\right|\left|\vec{\vec{\mathbf{V}}}\right|\left|\vec{\vec{\mathbf{R}}}^{T}\right| \\ &= \left|\vec{\vec{\mathbf{R}}}\right|\left|\vec{\vec{\mathbf{V}}}\right|\left|\vec{\vec{\mathbf{R}}}\right| \\ &= \left|\vec{\vec{\mathbf{V}}}\right|\left|\vec{\vec{\mathbf{R}}}\right|^{2} \\ &= \left|\vec{\vec{\mathbf{V}}}\right|\left(\pm 1\right)^{2} \\ &= \left|\vec{\vec{\mathbf{V}}}\right| \\ &> 0\end{aligned}$$

Since $\left|\vec{\vec{\boldsymbol{\Sigma}}}\right| > 0$ and $\left(\vec{\mathbf{x}}-\vec{\boldsymbol{\mu}}\right)^{T}\left(\vec{\mathbf{x}}-\vec{\boldsymbol{\mu}}\right) \geq 0$, then $\left(\vec{\mathbf{x}}-\vec{\boldsymbol{\mu}}\right)^{T}\vec{\vec{\boldsymbol{\Sigma}}}^{-1}\left(\vec{\mathbf{x}}-\vec{\boldsymbol{\mu}}\right) \geq 0$. Since $\left(\vec{\mathbf{x}}-\vec{\boldsymbol{\mu}}\right)^{T}\vec{\vec{\boldsymbol{\Sigma}}}^{-1}\left(\vec{\mathbf{x}}-\vec{\boldsymbol{\mu}}\right) \geq 0$, then $\dfrac{\left(\left(\vec{\mathbf{x}}-\vec{\boldsymbol{\mu}}\right)^{T}\vec{\vec{\boldsymbol{\Sigma}}}^{-1}\left(\vec{\mathbf{x}}-\vec{\boldsymbol{\mu}}\right)\right)^{i}}{2i} \geq 0$ for all $i = 1,2,3,\ldots$; hence, $-\dfrac{\left(\left(\vec{\mathbf{x}}-\vec{\boldsymbol{\mu}}\right)^{T}\vec{\vec{\boldsymbol{\Sigma}}}^{-1}\left(\vec{\mathbf{x}}-\vec{\boldsymbol{\mu}}\right)\right)^{i}}{2i} \leq 0$ for all $i = 1,2,3,\ldots.$ Furthermore, since $-\dfrac{\left(\left(\vec{\mathbf{x}}-\vec{\boldsymbol{\mu}}\right)^{T}\vec{\vec{\boldsymbol{\Sigma}}}^{-1}\left(\vec{\mathbf{x}}-\vec{\boldsymbol{\mu}}\right)\right)^{i}}{2i} \leq 0$ for all $i = 1,2,3,\ldots$ and $\exp\left(u\right) < 1$ for $u \in \mathbb{R}$ where $u < 0$, then we have the following:

$$y_{\substack{\text{scaled upper-half}\\ \text{rotated ellipsoid}}} = \frac{1}{\left(2\pi\right)^{\frac{n}{2}}\left|\vec{\vec{\mathbf{\Sigma}}}\right|^{\frac{1}{2}}}\exp\left(-\sum_{i=1}^{\infty}\frac{\left(\left(\vec{\mathbf{x}}-\vec{\boldsymbol{\mu}}\right)^{T}\vec{\vec{\mathbf{\Sigma}}}^{-1}\left(\vec{\mathbf{x}}-\vec{\boldsymbol{\mu}}\right)\right)^{i}}{2i}\right)$$

$$= \frac{1}{\left(2\pi\right)^{\frac{n}{2}}\left|\vec{\vec{\mathbf{\Sigma}}}\right|^{\frac{1}{2}}}\exp\left(\left(-\frac{\left(\left(\vec{\mathbf{x}}-\vec{\boldsymbol{\mu}}\right)^{T}\vec{\vec{\mathbf{\Sigma}}}^{-1}\left(\vec{\mathbf{x}}-\vec{\boldsymbol{\mu}}\right)\right)^{(1)}}{2\left(1\right)}\right)+\left(-\frac{\left(\left(\vec{\mathbf{x}}-\vec{\boldsymbol{\mu}}\right)^{T}\vec{\vec{\mathbf{\Sigma}}}^{-1}\left(\vec{\mathbf{x}}-\vec{\boldsymbol{\mu}}\right)\right)^{(2)}}{2\left(2\right)}\right)+\cdots\right)$$

$$= \frac{1}{\left(2\pi\right)^{\frac{n}{2}}\left|\vec{\vec{\mathbf{\Sigma}}}\right|^{\frac{1}{2}}}\exp\left(-\frac{\left(\left(\vec{\mathbf{x}}-\vec{\boldsymbol{\mu}}\right)^{T}\vec{\vec{\mathbf{\Sigma}}}^{-1}\left(\vec{\mathbf{x}}-\vec{\boldsymbol{\mu}}\right)\right)^{(1)}}{2\left(1\right)}\right)\exp\left(-\frac{\left(\left(\vec{\mathbf{x}}-\vec{\boldsymbol{\mu}}\right)^{T}\vec{\vec{\mathbf{\Sigma}}}^{-1}\left(\vec{\mathbf{x}}-\vec{\boldsymbol{\mu}}\right)\right)^{(2)}}{2\left(2\right)}\right)\cdots$$

$$\leq \frac{1}{\left(2\pi\right)^{\frac{n}{2}}\left|\vec{\vec{\mathbf{\Sigma}}}\right|^{\frac{1}{2}}}\exp\left(-\frac{\left(\left(\vec{\mathbf{x}}-\vec{\boldsymbol{\mu}}\right)^{T}\vec{\vec{\mathbf{\Sigma}}}^{-1}\left(\vec{\mathbf{x}}-\vec{\boldsymbol{\mu}}\right)\right)^{(1)}}{2\left(1\right)}\right)$$

Finally, since $y_{\substack{\text{scaled upper-half}\\ \text{rotated ellipsoid}}} \leq \frac{1}{\left(2\pi\right)^{\frac{n}{2}}\left|\vec{\vec{\mathbf{\Sigma}}}\right|^{\frac{1}{2}}}\exp\left(-\frac{\left(\left(\vec{\mathbf{x}}-\vec{\boldsymbol{\mu}}\right)^{T}\vec{\vec{\mathbf{\Sigma}}}^{-1}\left(\vec{\mathbf{x}}-\vec{\boldsymbol{\mu}}\right)\right)^{(1)}}{2\left(1\right)}\right)$ and

$y_{\substack{\text{pdf of the}\\ \text{multivariate}\\ \text{normal}}} = \frac{1}{\left(2\pi\right)^{\frac{n}{2}}\left|\vec{\vec{\mathbf{\Sigma}}}\right|^{\frac{1}{2}}}\exp\left(-\frac{1}{2}\left(\vec{\mathbf{x}}-\vec{\boldsymbol{\mu}}\right)^{T}\vec{\vec{\mathbf{\Sigma}}}^{-1}\left(\vec{\mathbf{x}}-\vec{\boldsymbol{\mu}}\right)\right)$, then we have the following:

$$y_{\substack{\text{scaled upper-half}\\ \text{rotated ellipsoid}}} \leq \frac{1}{\left(2\pi\right)^{\frac{n}{2}}\left|\vec{\vec{\mathbf{\Sigma}}}\right|^{\frac{1}{2}}}\exp\left(-\frac{\left(\left(\vec{\mathbf{x}}-\vec{\boldsymbol{\mu}}\right)^{T}\vec{\vec{\mathbf{\Sigma}}}^{-1}\left(\vec{\mathbf{x}}-\vec{\boldsymbol{\mu}}\right)\right)^{(1)}}{2\left(1\right)}\right)$$

$$= \frac{1}{\left(2\pi\right)^{\frac{n}{2}}\left|\vec{\vec{\mathbf{\Sigma}}}\right|^{\frac{1}{2}}}\exp\left(-\frac{1}{2}\left(\vec{\mathbf{x}}-\vec{\boldsymbol{\mu}}\right)^{T}\vec{\vec{\mathbf{\Sigma}}}^{-1}\left(\vec{\mathbf{x}}-\vec{\boldsymbol{\mu}}\right)\right)$$

$$= y_{\substack{\text{pdf of the}\\ \text{multivariate}\\ \text{normal}}}$$

Therefore, since $y_{\substack{\text{scaled upper-half}\\ \text{rotated ellipsoid}}} \leq y_{\substack{\text{pdf of the}\\ \text{multivariate}\\ \text{normal}}}$, then the probability density function of the multivariate Normal distribution centered at $\vec{\boldsymbol{\mu}} = \left(\mu_1,\ldots,\mu_n\right)$ with non-singular covariance $\vec{\vec{\boldsymbol{\Sigma}}}$ is bounded below by the second or higher approximation to the Taylor series expansion of the upper-half of the $n+1$ dimensional ellipsoid with identical covariance $\vec{\vec{\boldsymbol{\Sigma}}}$ centered at $\left(\mu_1,\ldots,\mu_n,0\right)$ and multiplied by the scalar $1\Big/\left((2\pi)^{n/2}\left|\vec{\vec{\boldsymbol{\Sigma}}}\right|^{1/2}\right)$.

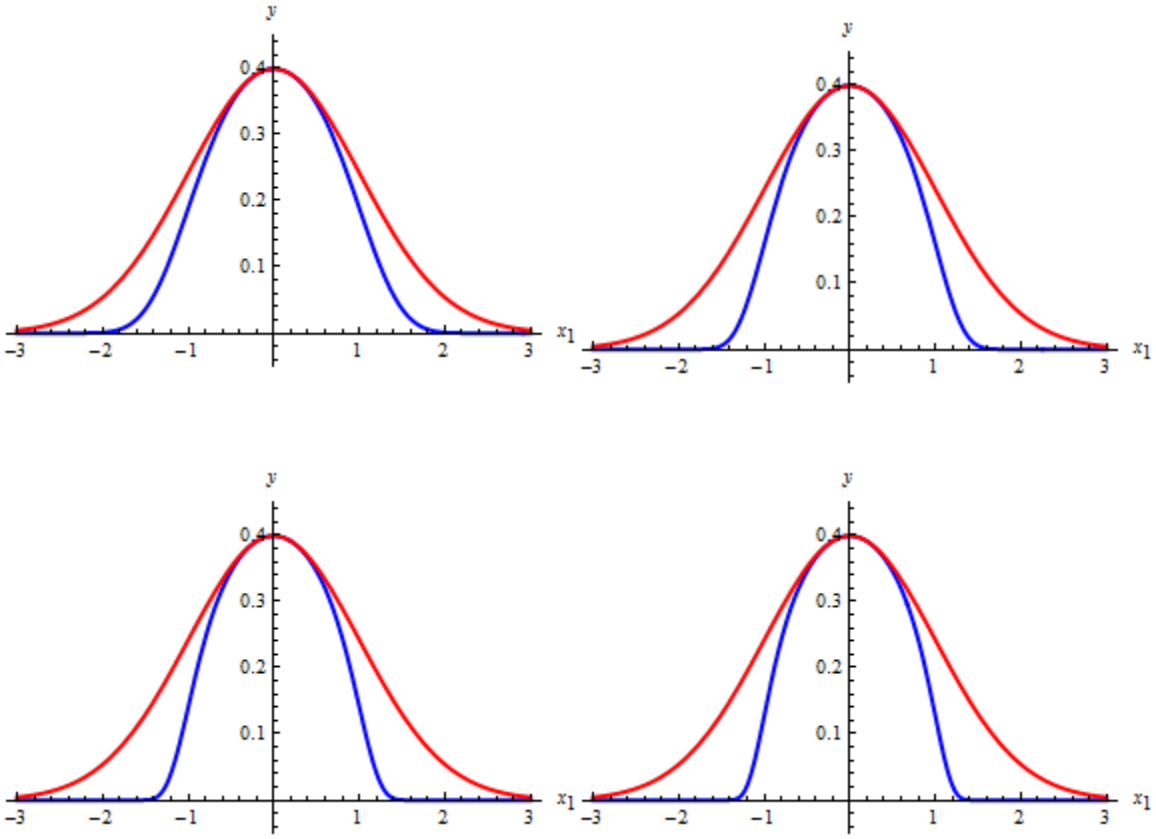
Example: Let x_1, y be variables aligned to the 2 axes of $\mathbb{R}^2$, then the equation for the probability density function of the multivariate Normal with $\mu_1 = 0$ and $\sigma_1 = 1$ is the following:

$$\begin{aligned} y_{\substack{\text{pdf of this}\\ \text{normal}}} &= \frac{1}{\left(2\pi\right)^{\frac{1}{2}}\left|\left[1\right]\right|^{\frac{1}{2}}}\exp\left(-\frac{1}{2}\left(\left[x_1\right]-\left[0\right]\right)^T\left[1\right]^{-1}\left(\left[x_1\right]-\left[0\right]\right)\right) \\ &= \frac{1}{\sqrt{2\pi}}\exp\left(-\frac{x_1^2}{2}\right) \end{aligned}$$

The scaled Taylor series expansion of the 2 dimensional ellipsoid is the following:

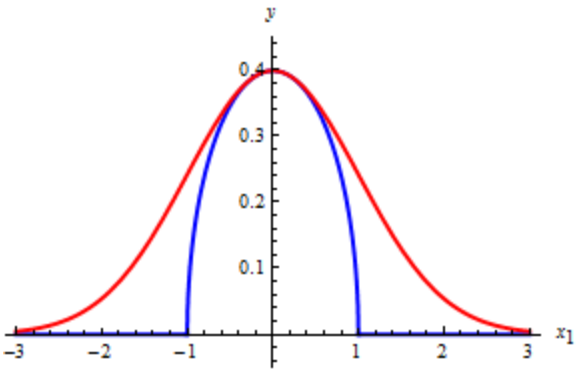
$$\begin{aligned} y_{\substack{\text{scaled upper-half}\\ \text{ellipsoid}}} &= \frac{1}{\left(2\pi\right)^{\frac{1}{2}}\left|\left[1\right]\right|^{\frac{1}{2}}}\exp\left(-\sum_{i=1}^{\infty}\frac{\left(\left(\left[x_1\right]-\left[0\right]\right)^T\left[1\right]^{-1}\left(\left[x_1\right]-\left[0\right]\right)\right)^i}{2i}\right) \\ &= \frac{1}{\sqrt{2\pi}}\exp\left(-\sum_{i=1}^{\infty}\frac{\left(x_1^2\right)^i}{2i}\right) \\ &= \frac{1}{\sqrt{2\pi}}\exp\left(-\sum_{i=1}^{\infty}\frac{x_1^{2i}}{2i}\right) \end{aligned}$$

If we plot the probability density function of this Normal on the same graph as the scaled second, third, fourth, and fifth approximations of the Taylor series expansion of the upper-half of the 2-dimensional ellipsoid, then we have the following:



where the red curve is the probability density function of this Normal and the blue curve is the second, third, fourth, and fifth approximations of the Taylor series expansion of the upper-half of the 2-dimensional ellipsoid, respectively.

Note that as we approach infinity, we will have the following:



Example: Let x_1, x_2, y be variables aligned to the 3 axes of $\mathbb{R}^3$, then the equation for the probability density function of the multivariate Normal with mean $\vec{\boldsymbol{\mu}} = \left(1,2\right)$, variances $\sigma_1^2 = \left(1\right)^2 = 1$ and $\sigma_2^2 = \left(3\right)^2 = 9$, and rotation $\theta = \frac{\pi}{3}$ is the following:

$$\begin{aligned}
\vec{\vec{\boldsymbol{\Sigma}}} &= \begin{bmatrix} \cos\left(\theta\right) & \cos\left(\theta + \frac{\pi}{2}\right) \\ \sin\left(\theta\right) & \sin\left(\theta + \frac{\pi}{2}\right) \end{bmatrix} \begin{bmatrix} \sigma_1^2 & 0 \\ 0 & \sigma_2^2 \end{bmatrix} \begin{bmatrix} \cos\left(\theta\right) & \cos\left(\theta + \frac{\pi}{2}\right) \\ \sin\left(\theta\right) & \sin\left(\theta + \frac{\pi}{2}\right) \end{bmatrix}^T \\
&= \begin{bmatrix} \cos\left(\frac{\pi}{3}\right) & -\sin\left(\frac{\pi}{3}\right) \\ \sin\left(\frac{\pi}{3}\right) & \cos\left(\frac{\pi}{3}\right) \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & 9 \end{bmatrix} \begin{bmatrix} \cos\left(\frac{\pi}{3}\right) & -\sin\left(\frac{\pi}{3}\right) \\ \sin\left(\frac{\pi}{3}\right) & \cos\left(\frac{\pi}{3}\right) \end{bmatrix}^T \\
&= \begin{bmatrix} 7 & -2\sqrt{3} \\ -2\sqrt{3} & 3 \end{bmatrix}
\end{aligned}$$

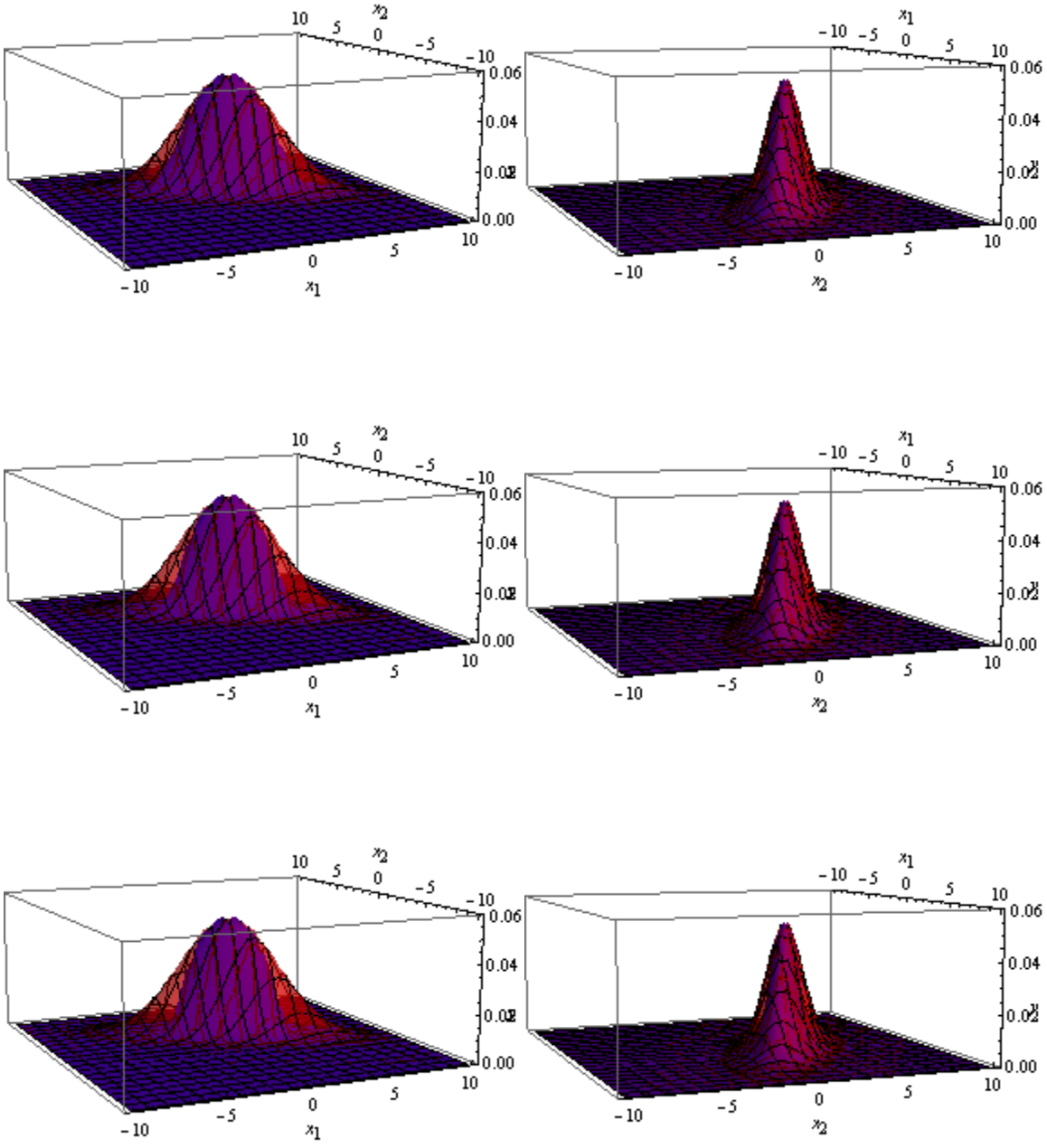
$$
\begin{aligned}
y_{\substack{\text{pdf of this}\\ \text{normal}}} &= \frac{1}{\left(2\pi\right)^{\frac{n}{2}}\left|\vec{\vec{\boldsymbol{\Sigma}}}\right|^{\frac{1}{2}}}\exp\left(-\frac{1}{2}\left(\vec{\mathbf{x}}-\vec{\boldsymbol{\mu}}\right)^{T}\vec{\vec{\boldsymbol{\Sigma}}}^{-1}\left(\vec{\mathbf{x}}-\vec{\boldsymbol{\mu}}\right)\right)\\
&= \frac{1}{\left(2\pi\right)^{\frac{2}{2}}\left\|\begin{bmatrix}7 & -2\sqrt{3}\\ -2\sqrt{3} & 3\end{bmatrix}\right\|^{\frac{1}{2}}}\exp\left(-\frac{1}{2}\left(\begin{bmatrix}x_1\\ x_2\end{bmatrix}-\begin{bmatrix}1\\ 2\end{bmatrix}\right)^{T}\begin{bmatrix}7 & -2\sqrt{3}\\ -2\sqrt{3} & 3\end{bmatrix}^{-1}\left(\begin{bmatrix}x_1\\ x_2\end{bmatrix}-\begin{bmatrix}1\\ 2\end{bmatrix}\right)\right)\\
&= \frac{1}{2\pi\left(9\right)^{\frac{1}{2}}}\exp\left(-\frac{1}{2}\begin{bmatrix}x_1-1\\ x_2-2\end{bmatrix}^{T}\begin{bmatrix}\frac{1}{3} & \frac{2}{3\sqrt{3}}\\ \frac{2}{3\sqrt{3}} & \frac{7}{9}\end{bmatrix}\begin{bmatrix}x_1-1\\ x_2-2\end{bmatrix}\right)\\
&= \frac{1}{6\pi}\exp\left(-\frac{1}{2}\begin{bmatrix}x_1-1\\ x_2-2\end{bmatrix}^{T}\begin{bmatrix}\frac{1}{3} & \frac{2}{3\sqrt{3}}\\ \frac{2}{3\sqrt{3}} & \frac{7}{9}\end{bmatrix}\begin{bmatrix}x_1-1\\ x_2-2\end{bmatrix}\right)
\end{aligned}
$$

The scaled Taylor series expansion of the upper-half of the 3-dimensional ellipsoid is the following:

$$
\begin{aligned}
y_{\substack{\text{scaled upper-half}\\ \text{rotated ellipsoid}}} &= \frac{1}{\left(2\pi\right)^{\frac{n}{2}}\left|\vec{\vec{\boldsymbol{\Sigma}}}\right|^{\frac{1}{2}}}\exp\left(-\sum_{i=1}^{\infty}\frac{\left(\left(\vec{\mathbf{x}}-\vec{\boldsymbol{\mu}}\right)^{T}\vec{\vec{\boldsymbol{\Sigma}}}^{-1}\left(\vec{\mathbf{x}}-\vec{\boldsymbol{\mu}}\right)\right)^{i}}{2i}\right)\\
&= \frac{1}{6\pi}\exp\left(-\sum_{i=1}^{\infty}\frac{\left(\left(\begin{bmatrix}x_1\\ x_2\end{bmatrix}-\begin{bmatrix}1\\ 2\end{bmatrix}\right)^{T}\begin{bmatrix}7 & -2\sqrt{3}\\ -2\sqrt{3} & 3\end{bmatrix}^{-1}\left(\begin{bmatrix}x_1\\ x_2\end{bmatrix}-\begin{bmatrix}1\\ 2\end{bmatrix}\right)\right)^{i}}{2i}\right)\\
&= \frac{1}{6\pi}\exp\left(-\sum_{i=1}^{\infty}\frac{\left(\begin{bmatrix}x_1-1\\ x_2-2\end{bmatrix}^{T}\begin{bmatrix}\frac{1}{3} & \frac{2}{3\sqrt{3}}\\ \frac{2}{3\sqrt{3}} & \frac{7}{9}\end{bmatrix}\begin{bmatrix}x_1-1\\ x_2-2\end{bmatrix}\right)^{i}}{2i}\right)
\end{aligned}
$$

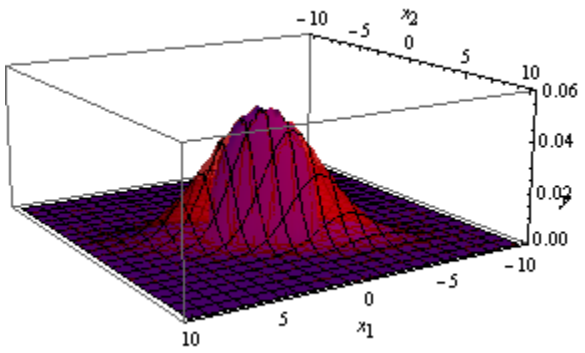
If we plot the probability density function of this Normal on the same graph as the scaled second, third, and fourth approximations of the Taylor series expansion of the

upper-half of the 3-dimensional ellipsoid, then we have the following (two perspective plots per approximation with the second to fourth approximation displayed from top to bottom):



where the outer red curve is the probability density function of this Normal and the inner blue curve is the second, third, and fourth approximations of the Taylor series expansion of the upper-half of the 3-dimensional ellipsoid, respectively from top to bottom.

Note that as we approach infinity, we will have the following:



where the outer red curve is the probability density function of this Normal and the inner blue curve is the Taylor series expansion of the upper-half of the 3-dimensional ellipsoid as its approximation approaches infinity.

LIST OF REFERENCES

Alpaydin, E. (2004). *Introduction to machine learning (adaptive computation and machine learning)* The MIT Press.

Bishop, C. M. (2007). *Pattern recognition and machine learning (information science and statistics)* (1st ed.) Springer.

Bradski, G., & Kaehler, A. (2008). *Learning OpenCV: Computer vision with the OpenCV library* (1st ed.) O'Reilly Media, Inc.

Burden, R. L., & Faires, J. D. (2005). *Numerical analysis* (8th ed.). Belmont, CA: Thomson Brooks/Cole.

Duda, R. O., Hart, P. E., & Stork, D. G. (2000). *Pattern classification (2nd edition)* (2nd ed.) Wiley-Interscience.

Hastie, T., Tibshirani, R., & Friedman, J. H. (2001). *The elements of statistical learning : Data mining, inference, and prediction : With 200 full-color illustrations*. New York: Springer.

Izenman, A. J. (2008). *Modern multivariate statistical techniques : Regression, classification, and manifold learning*. New York; London: Springer.

Mahalanobis, P. C. (1936). On the generalised distance in statistics *Proceedings of the National Institute of Sciences of India, 2*(1), 49–55.

MNIST Handwritten Digit Database, Yann LeCun and Corinna Cortes. Retrieved 9/28/2009 from http://yann.lecun.com/exdb/mnist/

OpenCV 1.1 (2008). *Open Computer Vision Library Downloads.*

Parzen, E. (1962). On estimation of a probability density function and mode. *The Annals of Mathematical Statistics, 33*(3), 1065–1076.

Random Forests. Retrieved 9/28/2009 from http://www.stat.berkeley.edu/~breiman/RandomForests/cc_home.htm#overview

Samet, H. (2006). *Foundations of multidimensional and metric data structures*. Amsterdam ; Boston: Elsevier/Morgan Kaufmann.

UCI Machine Learning Repository: Breast Cancer Wisconsin (Diagnostic) Data Set. Retrieved 9/28/2009 from http://archive.ics.uci.edu/ml/datasets/Breast+Cancer+Wisconsin+(Diagnostic)

Vincent, P., & Bengio, Y. (2002). Manifold parzen windows. *Advances in Neural Information Processing Systems 15*, 825-832.

Wang, L., Bo, L., & Jiao, L. (2006). Rough sets and knowledge technology; A modified K-means clustering with a density-sensitive distance metric., 544.

Wasserman, L. (2007). *All of nonparametric statistics (springer texts in statistics)* Springer.

Zezula, P. (2006). *Similarity search : The metric space approach*. New York: Springer.

INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center
   Ft. Belvoir, Virginia

2. Dudley Knox Library
   Naval Postgraduate School
   Monterey, California

3. Kevin Squire
   Naval Postgraduate School
   Monterey, California

4. Mathias Kolsch
   Naval Postgraduate School
   Monterey, California

5. John Falby
   Naval Postgraduate School
   Monterey, California